UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior - Leganés

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

GiDtoNet: Interfaz de Preproceso para el Mallado de Entidades Geométricas. Comparación entre los Malladores *GiD*® y *NetGen*.

AUTORA: Laura Vozmediano Latorre.
TUTOR: Daniel García Doñoro.
DIRECTOR: Luis Emilio García Castillo.

Leganés, 2010

Título: *GiDtoNet*: Interfaz de Preproceso para el Mallado de Entidades Geométricas. Comparación entre los Malladores *GiD*® y *NetGen*.

Autor: *Laura Vozmediano Latorre.*

Tutor: *Daniel García Doñoro.*

Director: *Luis Emilio García Castillo.*

La defensa de este Proyecto Fin de Carrera se realizó el día 28 de octubre de 2010; siendo calificada por el siguiente tribunal:

Presidente: Magdalena Salazar Palma.

Secretario: Sergio Llorente Romano.

Vocal: Ignacio Gómez Revuelto.

Habiendo obtenido la siguiente calificación:

CALIFICACIÓN:

Presidente **Secretario** **Vocal**

A mis padres y mi hermana Raquel.
A Jose.

Agradecimientos

A lo largo de los 7 años que he pasado en la universidad, he tenido momentos muy buenos y alguno un poco peor, aunque a mí siempre me gusta quedarme con las cosas positivas. Lo importante es que al final todo llega y con la presentación del Proyecto Fin de Carrera pongo punto y final a una etapa llena de nuevas experiencias, mucho trabajo y una gran dedicación y esfuerzo.

Durante mi etapa universitaria he podido conocer a mucha gente, algunos de ellos se han acabado convirtiendo en grandes amigos gracias a la complicidad que proporciona el hecho de compartir experiencias similares. Alberto, gracias por decirme cada día lo mucho que valgo y por hacerme ver la vida desde otro punto de vista. Siempre has estado cerca para ayudarme, sin importar cómo ni cuándo. Sabes que eres uno de mis mejores amigos. Patricia, Pablo, Alicia, César, muchas gracias por entenderme, ayudarme a superar situaciones complicadas y conseguir que me tome la vida con más calma. El mejor recuerdo que tengo de todos vosotros es haber recorrido el camino juntos, sufriendo por alcanzar el mismo objetivo.

A pesar de haber encontrado alguno de mis mejores amigos entre mis compañeros de la universidad, no me puedo olvidar de aquéllos con los que he crecido. Todos han sabido darme ánimo esos días en los que no sabes porqué estás haciendo lo que haces y buenos consejos cuando he tenido que tomar decisiones importantes en mi vida, como mi primer trabajo. Raquel, Cristina, Eric, Víctor, Ángel, gracias a todos.

De todos ellos, hay una persona que es especial. Con ella he compartido largas mañanas de atascos, desayunos interminables en la cafetería de la universidad y los momentos más duros y felices que he podido vivir hasta hoy. Gracias Elena por darme ánimo y estar cerca siempre que te he necesitado. Te mereces todo lo bueno que te pase.

Estoy segura de que la vida nos depara grandes cosas a todos. Sois los mejores amigos que una persona podría tener.

A parte de grandes amigos, mi paso por la universidad ha dejado en mi vida a la persona más importante de todas las que me rodean. Gracias a tí, Jose, he conseguido alcanzar todas mi metas, me has ayudado, apoyado, aconsejado y animado. Gracias por estar a mi lado siempre y por llenar los últimos 6 años de mi vida de recuerdos inolvidables. Sin tí no lo habría conseguido. Quiero seguir recorriendo el camino contigo.

Por último, pero no menos importante, quiero agradecer a toda mi familia la confianza depositada en mí, ellos sabían que lo conseguiría. A mi hermana Raquel, gracias por escucharme y ser una gran aliada. Ahora que no estás en Madrid me doy cuenta de lo que te echo de menos. Y por supuesto a mis padres, que gracias a ellos me he convertido en la persona que soy. Todo lo que he hecho a lo largo de mi vida ha sido siempre para que estéis orgullosos de mí y si he llegado hasta aquí ha sido gracias a vuestro apoyo y a que me habéis dejado que sea yo la que tome mis propias decisiones.

Ahora que finalizo esta etapa y comienzo mi vida 'adulta' espero poder compartir los próximos años con todos vosotros, porque lo más importante en la vida de una persona es la gente que la rodea y yo en eso soy una afortunada.

Muchas gracias a todos por permanecer siempre ahí.

Muchas gracias a mi tutor y mi director de Proyecto, Daniel García Doñoro y Luis Emilio García Castillo, por el tiempo que me han dedicado y por estar siempre disponibles para cualquier duda o problema que me surgiera.

Laura.

Resumen

Este Proyecto Fin de Carrera presenta una interfaz basada en el preprocesador *GiD*® que permite generar el fichero de entrada del generador de mallado *NetGen*. *GiD*® proporciona el soporte necesario para crear las entidades geométricas soportadas por *NetGen*. Además, se realiza una comparación entre las mallas generadas por ambos generadores de mallado bajo las mismas opciones de configuración.

La interfaz se ha desarrollado para tener una herramienta que permita realizar un estudio entre las propiedades de los mallados generados con *GiD*® y los generados con *NetGen*. Hay que tener en cuenta que esta comparación se puede desarrollar gracias a que ambos generadores de mallado trabajan bajo la misma configuración, es decir, se comportan de la misma forma a la hora de crear los mallados.

La creación de estructuras, las operaciones que se pueden realizar entre ellas y la asignación de las condiciones de contorno se realizan mediante las diferentes opciones que presenta la interfaz por medio de ventanas o *TKWidgets* (Ventanas desarrolladas en lenguaje Tcl-Tk). Sin embargo, para eliminar alguna de las entidades geométricas estes el usuario debe utilizar las herramientas que proporciona *GiD*® y que están completamente integradas con la interfaz. Una vez que el usuario ha definido las estructuras que quiere plasmar en su proyecto, la interfaz ejecuta varios algoritmos implementados en lenguaje Tcl-Tk para obtener la información geométrica de todas ellas.

Todas las funcionalidades del módulo implementado hacen posible que finalmente realicemos una comparativa entre los mallados sobre diversas estructuras proporcionados por *GiD*® y *NetGen* para saber cuál de los dos presenta mejores propiedades. Esta comparación se realiza mediante los parámentros que proporciona *GiD*® para medir las calidades de los mallados. Este estudio se presentará al final de este Proyecto Fin de Carrera.

Abstract

This work presents an interface based on the *GiD*® preprocesor that generates the input file for the *NetGen* mesh generator. *GiD*® provides the support necessary to create geometric entities supported by *NetGen*. In addition, it performs a comparison between the meshes generated by both mesh generators under the same settings.

The interface has been developed to have a tool that allows a study of the properties of generated meshes generated GID and NET. Keep in mind that this comparison can be developed through both mesh generators operating under the same configuration, ie, they behave the same way when creating the meshes. The interface has been developed to have a tool that allows a study of the properties of meshes generated with *GiD*® and *NetGen*. We have to consider that this comparison can be done because both mesh generators have the same configuration, they create meshes by the same way.

The structures creation, the operations between two of them and the boundary conditions assignment are made using different interface options presented through windows or *TKWidgets* (windowsdeveloped in Tcl-Tk). However, to eliminate some of the geometric entities present in the project, the user must use the tools provided by *GiD*® that are fully integrated with the interface. Once the user has defined the structures of his project, the interface implements several algorithms implemented in Tcl-Tk to obtain the geometric information of all of them.

All features of the implemented module can perform a comparison between the meshes of several structures provided by *GiD*® and *NetGen* to know which of the two meshers has better properties. This comparison is done through providing *GiD*® Parameters for measuring the qualities of the meshes. This study will be presented at the end of this document.

Índice general

Índice de figuras

Índice de pseudocódigos

Capítulo 1

Introducción

1.1. Antecedentes.

Las tecnologías actuales en el diseño de estructuras mecánicas, electromagnéticas o de cualquier otro campo requieren el cálculo con precisión de sus parámetros característicos por su importancia en el diseño final. Durante los últimos años, una gran variedad de geometrías, materiales y configuraciones han sido usadas en su diseño, por lo que es necesario contar con una herramienta lo suficientemente potente para realizar el análisis de estructuras complicadas y, a la vez, flexible para poder aplicarla a una extensa variedad de configuraciones.

Los métodos numéricos son técnicas mediante las cuales es posible formular problemas de tal forma que sean resueltos con operaciones aritméticas. Detrás del análisis de cualquier problema físico se realiza previamente un modelo matemático de este modelo físico. Los métodos numéricos realizan una proyección del problema en un espacio finito. A este proceso se le denomina discretización del problema continuo. Mediante esta discretización pasamos de tener un problema en una región contínua a tenerlo en una región discreta.

Como consecuencia de esta discretización se obtiene un conjunto de ecuaciones algebraicas, en las que el número de incógnitas dependerá cómo se realice dicha discretización. Existen muchas formas de elegir este conjunto de funciones que forman una base vectorial sobre la que se aproxima la solución exacta del problema. Una vez obtenido este conjunto de ecuaciones se resuelve el sistema y se obtiene la solución. Esta solución es la que caracteriza el comportamiento del problema físico que se ha estudiado. Finalmente se puede realizar un postproceso de los resultados obtenidos.

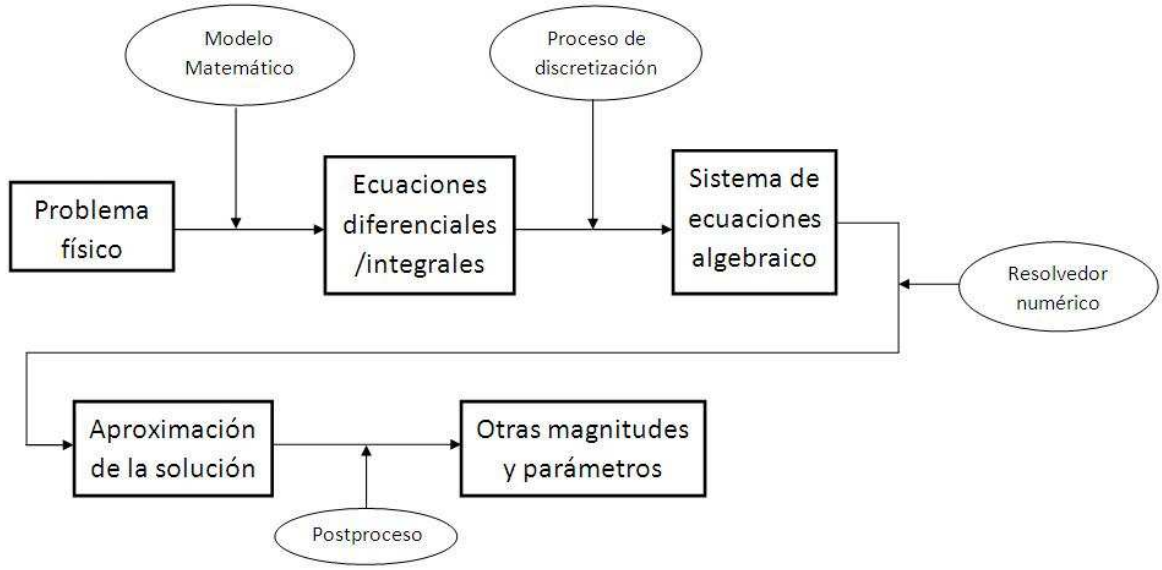
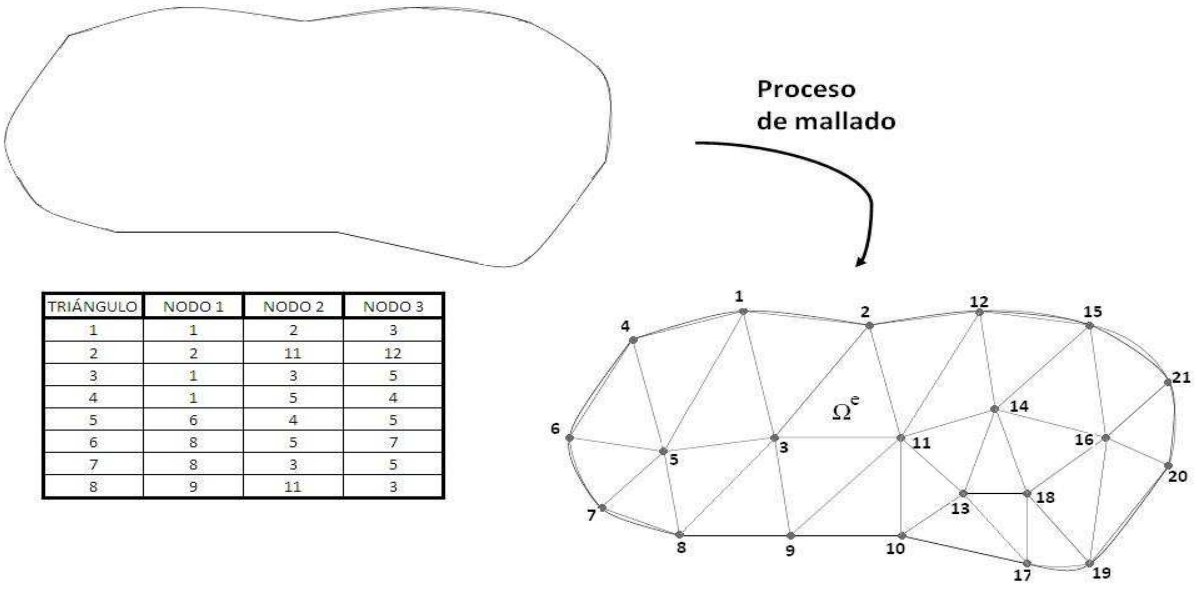


Figura 1.1: Análisis de un problema físico por un método numérico.

Existen una gran cantidad de métodos numéricos destinados a la búsqueda de la solución a un problema físico. En función del problema que se quiere estudiar se utilizará uno u otro. Algunos ejemplos de métodos numéricos son el Método de las Diferencias Finitas (*Finite Difference Method, FD*), el Método de los Momentos (*the Method of moments, MoM*) y el Método de los Elementos Finitos (*Finite Elements Method, FEM*).

Como se ha visto anteriormente, para poder analizar correctamente un problema físico, es necesario realizar una discretización por subdominios del problema. Debido a la necesidad de obtener esta dicretización del problema que se está planteando, existen unas herramientas que se encargan de realizar mallados denominados generadores de mallado.

Un generador de mallado es una herramienta que permite, mediante una serie de operaciones, realizar la división del dominio de un problema en un conjunto de subdominios. Cada uno de estos subdominios se encuentra "conectado" con los subdominios vecinos mediante una serie de puntos (denominados nodos). Es importante destacar que cuanto más fino sea el mallado y más elementos lo formen, más grande será el sistema de ecuaciones que se debe resolver. Toda la información sobre la "conectividad" entre los diferentes elementos que forman la malla está disponible siempre gracias a la exportación de los datos desde el propio generador de mallado. En la figura 1.2 se muestra un ejemplo de la discretización del dominio de un problema físico. En este caso el generador de mallado ha realizado una división en subdominios mediante una malla formada por triángulos.



TRIÁNGULO	NODO 1	NODO 2	NODO 3
1	1	2	3
2	2	11	12
3	1	3	5
4	1	5	4
5	6	4	5
6	8	5	7
7	8	3	5
8	9	11	3

Figura 1.2: Ejemplo de la discretización de un dominio irregular.

Existen una gran variedad de generadores de mallado en el mercado, algunos de ellos necesitan licencia para poder ser utilizados y otros, los de software libre, los aportan sus creadores para llevar a cabo estudios minuciosos de estructuras que serían difíciles de estudiar si no dispusiéramos de un generador de mallado. La mayoría de estos generadores de mallado trabajan con estructuras en 2 dimensiones (2D) y 3 dimensiones (3D) pudiendo, de esta forma, analizar prácticamente cualquier tipo de problema. El diseño de la estructura corre a cargo del usuario que, mediante las herramientas que proporcione cada generador de mallado, debe construir de la manera más fiable posible la estructura que quiere estudiar.

La existencia de un gran número de generadores de mallado lleva a los usuarios a preguntarse cuál de ellos debe utilizar en cada momento, ya que cada uno posee una serie de propiedades que lo caracterizan. Algunos de los generadores de mallado que podemos encontrar son:

- *Cubit*. Es una herramienta software de mallado en 2D y 3D. Su objetivo principal es diminuir el tiempo de creación de mallas. Se utiliza principalmente para realizar la discretización de dominios y resolverlos con el Método de los Elementos Finitos. *Cubit* contiene muchos algoritmos para controlar y automatizar la mayor parte del proceso de mallado [1].

- *GiD*®. Es un generador de malla comercial de los muchos que existen en el mercado. Permite la realización de pre y postproceso de estructuras en 2D y 3D. Tiene una gran capacidad de personalización porque permite añadir nuevas opciones a sus menús para cubrir las necesidades del usuario. *GiD*® es un generado de mallado muy importante

porque genera una discretización del dominio del problema para que pueda ser resuelto por cualquier método numérico [2].

- *Gmsh*. Es un generador de malla de elementos finitos en 3D. Dispone además de un módulo de post-proceso. El objetivo de *Gmsh* es proporcionar ua herramienta de mallado rápida, de uso fácil y con capacidades avanzadas de visualización. *Gmsh* se basa en cuatro módulos: geometría, malla, solucionador y post-procesamiento. La especificación de cualquier entrada a estos módulos se realiza de forma interactiva mediante la interfaz gráfica de usuario o en archivos de texto ASCII [3].

- *MeshGen*. Herramienta que genera mallas en 2D mediante triángulos y cuadriláteros. El formato de salida del fichero que proporciona la información del mallado es compatible con la resolución de la estructura mediante el Método de los Elementos Finitos. Para más información consultar su página web [4].

- *NetGen*. Generador automático de malla tetraédrica en 3D. La geometría del problema se introduce a partir de un fichero con una estructura determinada. Gracias a la capacidad que tiene de crear mallados mediante tetraedros se utiliza para discretizar el dominio en problemas que se resolverán mediante el Método de los Elementos Finitos [5].

- *TrueGrid*®. Es un generador de mallado comercial en 2D y 3D destinado al desarrollo de análisis *FEA (Finite Element Analysis)* y *CFD (Computational Fluid Dynamics)*. Los usuarios tienen el control completo sobre el diseño de la malla. Todos los mallados son estructurados. Para más información consultar su página web [6].

Para mayor información sobre los distintos tipos de malladores se puede consultar la página web *http://www.andrew.cmu.edu/user/sowen/softsurv.html* en la que aparece un listado con cada uno de ellos y la posibilidad de acceder a todas sus características.

Después de enumerar algunos de los muchos generadores de mallado que existen, vamos a centrarnos en estudiar uno de ellos, *NetGen*, que como ya se ha comentado anteriormente genera mallas tetraédricas en 3D y que fue creado a partir de un estudio basado en la resolución de estructuras mediante el Método de los Elementos Finitos.

La introducción de la geometría de un problema físico en *NetGen* se realiza mediante la carga de un fichero con una serie de primitivas que se detallarán en la sección 4.2. Este fichero puede ser generado a mano, pero para problemas de ingeniería complejos, esta forma es muy lenta y costosa, con lo que, el objetivo principal que se busca es obtener dicho fichero de manera automática. Si se consigue este objetivo, la reducción de tiempo para el usuario es considerable, ya que la generación manual del fichero con la información geométrica y topológica correctas de estructuras complejas puede llevar días, y en cambio, si se hiciera de manera automática llevaría minutos.

Como se ha descrito en el listado anterior, *GiD*®, es un software comercial muy útil que permite la personalización de su interfaz gráfica añadiendo nuevos menús con las funcionalidades

que el usuario puede necesitar. Esta característica es por la que hemos elegido la herramienta *GiD* para desarrollar el presente Proyecto Fin de Carrera. Se va a incluir un menú que permitirá la creación automática por parte de *GiD* del fichero de entrada a *NetGen* con la geometría del problema desarrollado, haciendo mucho más fácil y rápido para el usuario la interacción con *NetGen* ya que se evita que construya este fichero manualmente.

Aprovechando otra de las características de *GiD*, se va a realizar un estudio (a lo largo del capítulo 5) sobre las cualidades de una malla creada por este generador de mallado y otra creada por *NetGen*, que estará disponible en *GiD*, para averiguar cuál de las dos presenta mejores resultados bajo las mismas condiciones. Esta comparativa se puede realizar gracias a la herramienta que proporciona *GiD* para estudiar las cualidades de un mallado, denominada *Mesh Quality*.

1.2. Objetivos.

Como se ha comentado anteriormente, *GiD*® tiene una interfaz de usuario que permite desarrollar las estructuras de los problemas gráficamente. Este Proyecto Fin de Carrera se apoya en *GiD*® y proporciona un módulo que permite generar el fichero que necesita *NetGen* para cargar la geometría de un problema físico determinado. De esta forma el usuario de *NetGen* es capaz de generar la geometría de su problema de una forma rápida y sencilla.

El módulo desarrollado en este Proyecto Fin de Carrera permite que el usuario tenga a su disposición el mallado generado por *NetGen* de la estructura que está estudiando en la interfaz gráfica de *GiD*. Esta malla se crea gracias a una llamada remota a *NetGen*.

Finalmente y gracias a una serie de opciones de las que dipone *GiD*®, este Proyecto Fin de Carrera realiza un estudio de la malla obtenida mediante el generador de mallado *NetGen* y la malla que *GiD*® generaría para el mismo caso y bajo las mismas condiciones. Con este estudio se pretende comprobar cuál de las dos herramientas de mallado posee mejores características y para qué casos.

1.3. Contenido del proyecto.

La estructura que presenta este Proyecto Fin de Carrera es la siguiente:

- **Capítulo 1.**

  Contiene una breve explicación de los motivos por los que se ha llevado a cabo este Proyecto Fin de Carrera (Sección 1.1), así como una breve descripción de los objetivos marcados (Sección 1.2).

- **Capítulo 2.**

  Presentación muy detallada de las herramientas software utilizadas para la realización de este Proyecto Fin de Carrera. Se describen las principales características y funcionalidades del *pre-post procesador* de propósito general *GiD*® (Sección 2.1) y las del segundo generador de mallado bajo estudio, *NetGen* (Sección 2.2). Finalmente se presenta una comparativa con similitudes y diferencias entre ambos generadores de mallado (Sección 2.3).

- **Capítulo 3.**

  Se describen las características que posee el módulo implementado en este Proyecto Fin de Carrera y se explican los pasos que se deben seguir para desarrollar un proyecto con el módulo *GiDtoNet* (Sección 3.1). Estos pasos son los siguientes: inicializar el módulo (Sección 3.2), crear las entidades geométricas (Sección 3.3), modificar en cualquier momento el proyecto (Sección 3.4), asignar las condiciones de contorno (Sección 3.5), gestionar las operaciones que se realicen (Sección 3.6), crear el fichero *CSG* (Sección 3.7) y por último crear el fichero que contiene la información del mallado (Sección 3.8).

- **Capítulo 4.**

  Este capítulo está destinado a que el lector comprenda la finalidad y la estructura de todos los ficheros involucrados en el módulo *GiDtoNet*. Estos ficheros son *GiD2Net.geo* (Sección 4.2), el *GiDtoNet.cnd* (Sección 4.3), el *script.bat* (Sección 4.4) y por último los de mallados *GiD2Net.msh* y *Net2GiD.msh* (Sección 4.5).

- **Capítulo 5.**

  Presentación de las pruebas realizadas observando cuál de los dos generadores de mallado posee mejores propiedades bajo la misma configuración. Los mallados llevados a cabo son de una esfera (Sección 5.2), un cilindro (Sección 5.4), un hexaedro (Sección 5.3), un cono (Sección 5.5), de una antena tipo bocina (Sección 5.6) y, por último, un Hércules C-130 (Sección 5.7). Todas las pruebas están acompañadas de una serie de gráficas con los resultados obtenidos que apoyan las conclusiones planteadas.

- **Capítulo 6.**

  Conclusiones finales del estudio realizado en este Proyecto Fin de Carrera (Sección 6.1) y sugerencias para continuar la investigación en esta línea (Sección 6.2).

- **Capítulo 7.**

  En este último, se incluye un breve presupuesto de los gastos que ocasionaría el desarrollo de este Proyecto Fin de Carrera si lo hubiera realizado un ingeniero titulado.

Capítulo 2

Herramientas Software Utilizadas

A lo largo de este capítulo, se realizará una descripción detallada de las principales características y funcionalidades de las herramientas software que están involucradas en este Proyecto Fin de Carrera. Por un lado tenemos el *pre-post procesador* de propósito general *GiD*® y por otro el generador de mallado *NetGen*.

2.1. *Pre-post procesador* de propósito general *GiD.*

2.1.1. Descripción general.

GiD® es una interfaz gráfica de usuario desarrollado por el *Internation Center for Numerical Methods in Engineering (CIMNE)* [7]. Fue creada para cubrir todas las necesidades que pudieran existir en cuanto a simulaciones numéricas mediante el preprocesado y el postprocesado de los problemas físicos que el usuario quiera estudiar. *GiD*® está destinado, por tanto, a la definición y preparación de los datos necesarios para realizar esas simulaciones numéricas (siempre externas a *GiD*®), así como la visualización de los resultados obtenidos [1].

Al tratarse de una interfaz gráfica de propósito general, permite importar y exportar la geometría o el mallado de las estructuras en la mayoría de los formatos usados por los principales software comerciales de *CAD* (*Computer Aided Design*). Así, algunos de los formatos soportados por *GiD*® son *IGES*, *ACIS*, *DXF*, *VRML*, *3D Studio* o *Rhinoceros* por citar algunos. Además, posee una gran capacidad multi-lenguaje que permite traducir a varios idiomas los menús, cuadros de diálogo y ventanas de los módulos o problemas tipo desarrollados [8].

GiD® tiene una gran flexibilidad y capacidad de personalización. Esta personalización se lleva a cabo mediante módulos o problemas tipo (*problem type* en terminología anglosajona) que pueden ser implementados por los desarrolladores realizando una definición específica de algunas de las propiedades de las geometrías que se pueden crear. Algunas de las acciones que *GiD*® soporta a la hora de crear un problema tipo son la introducción de nuevas barras de herramientas para facilitar el acceso a los menús o la eliminación de algunas opciones que el usuario no va a utilizar. La implementación de todas estas opciones que *GiD*® tiene a disposición de los desarrolladores pueden llevarse a cabo gracias al lenguaje de programación Tcl-Tk.

El lenguaje de programación *Tcl* (*Tool Command Language*), es un lenguaje de script creado por John Ousterhout, que ha sido concebido con una sintaxis sencilla para facilitar su aprendizaje, sin perder funcionalidad y expresividad. Se utiliza principalmente para el desarrollo rápido de prototipos, aplicaciones tipo script, interfaces gráficas y pruebas. La combinación de Tcl con Tk (*Tool Kit*) es conocida como Tcl-Tk, y se utiliza para la creación de interfaces gráficas [9].

Todos los tipos de estructuras utilizadas para la visualización de los resultados procedentes de las simulaciones numéricas están presentes en el módulo de postprocesado de *GiD*®, tales como líneas de contorno, gráficos vectoriales, diagramas, líneas de flujo, deformaciones, etc.

A modo de ejemplo, se muestra a continuación un modelo generado con la herramienta *GiD*®. En la figura 2.1 se muestra la geometría de una estructura creada mediante las entidades disponibles en *GiD*®: puntos, líneas, superficies y volúmenes. En la figura 2.2 se aprecia el mallado llevado a cabo por *GiD*®. Cada uno de estos elementos actuará como subdominio a la hora de realizar simulaciones numéricas.

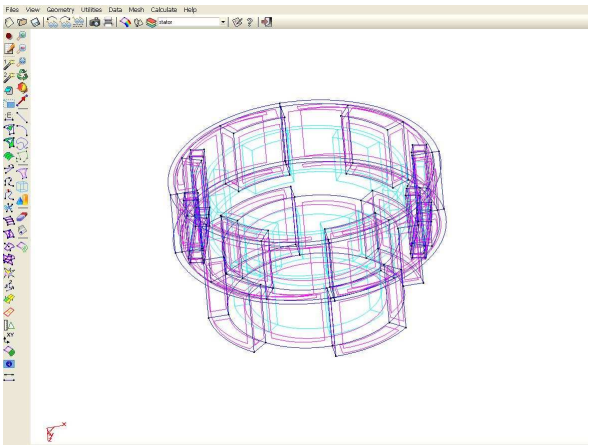


Figura 2.1: Preprocesado de un motor.

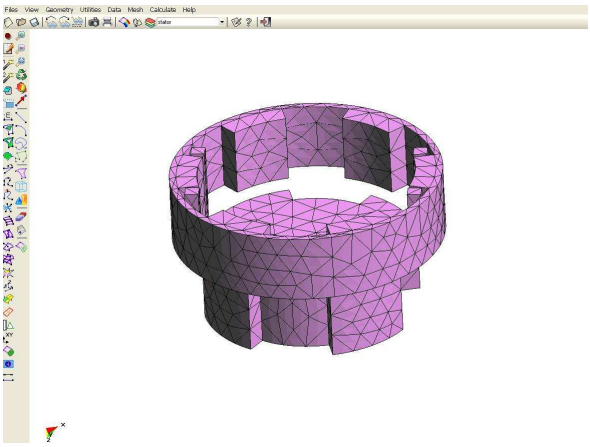


Figura 2.2: Mallado de un motor.

Además se incluye en la figura 2.3 un ejemplo de la visualización de los resultados que se puede llevar a cabo con *GiD*®. En este caso, este problema consiste en una pieza dentada en la que se puede ver el resultado obtenido tras realizar la resolución de la estructura mediante algún método numérico.

Si se desea obtener más información sobre *GiD*® puede descargarse el manual de usuario, el manual de referencia, numerosos tutoriales o presentaciones de su Web [10].

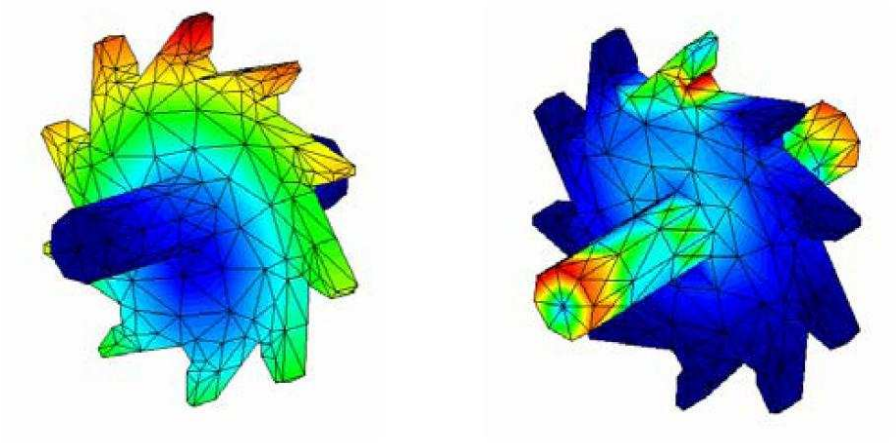


Figura 2.3: Postproceso de un problema realizado con *GiD*.

2.1.2. Generación del mallado mediante *GiD*.

GiD® permite la generación de mallados (mediante elementos lineales y cuadráticos) de un modo rápido y eficiente usando un módulo de mallado para volúmenes y superficies siguiendo dos tipos de criterios diferentes:

- El mallado estructurado se realiza mediante triángulos, cuadriláteros, hexaedros, hexaedros y tetraedros.

- El mallado no estructurado es generado automáticamente. Está basado en un criterio de calidad y espaciado definido por el usuario. Se puede generar un mallado formado por muchos tipos de elementos: triángulos, cuadriláteros, círculos, esferas y tetraedros.

Existen además tres malladores de superficie no estructurados para crear mallados mediante triángulos o cuadriláteros. Están basados en la técnica de avanzado frontal, mediante la que se genera el mallado desde la superficie hacia dentro del volumen.

- RFAST. Genera la malla de la superficie (2D) y mapea el resultado en las tres dimensiones del espacio. La malla se genera muy deprisa pero la calidad de los resultados no es tan buena en algunos casos.
- RSURF. Genera la malla directamente en 3D. Es mucho más lento que RFAST pero proporciona mejores resultados.
- RJUMP. Genera la malla directamente en 3D de un grupo de superficies saltándose las líneas que conectas dichas superficies. El orden en el que se realiza el mallado sigue el siguiente criterio: tangencia entre superficies vecinas, líneas seleccionadas por el usuario y continuidad entre superficies de curvatura.

En la figura 2.4 se muestra el menú de configuración que *GiD*® proporciona a sus usuarios para que establezcan las opciones que consideren oportunas en función del tipo de mallado que quieran crear.

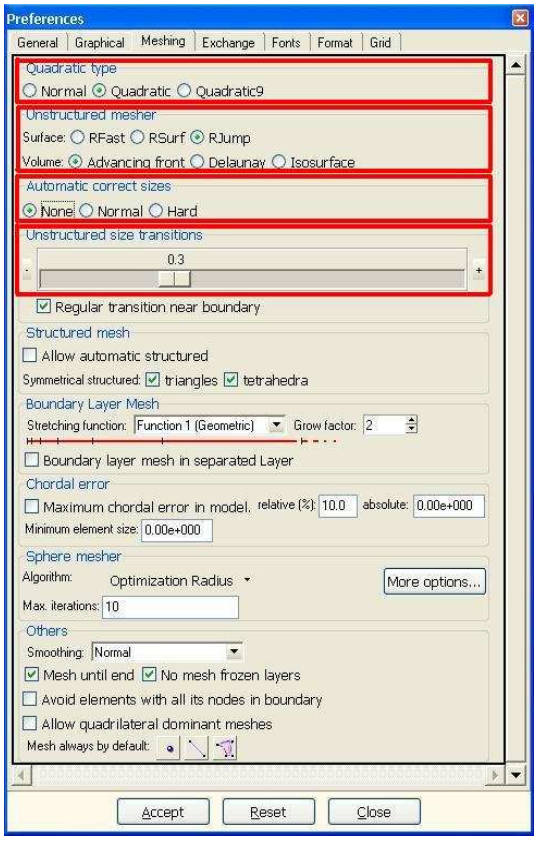


Figura 2.4: Menú de configuración para el mallado con *GiD*.

En nuestro caso las opciones de configuración del generador de mallado que se van a utilizar son las siguientes:

- ***Quadratc Type***. Permite seleccionar el grado de los elementos con los que se va a realizar el mallado. Si se selecciona *Normal* se proporcionará un nodo por vértice del elemento de mallado. Con *Quadratic* tenemos nodos en cada uno de los vértices del elemento y en los puntos medios de sus líneas y *Quadratic9* proporciona un nodo extra en el centro de los cuadrados con los que malla.

- ***Unstructured mesher***.

  - *Surface*. Esta opción permite seleccionar el tipo de mallado superficial que queremos que se construya. Las opciones que existen, como acabamos de comentar son: RFast, RSurf y RJump.
  - *Volumen*. La opción *Advancing front* para crear mallados en los que los tetraedros se creen desde las superficies hacia dentro de los volúmenes.

- ***Automatic correct sizes***. La opción *None* permite que el tamaño de mallado que imponga el usuario para la estructura sea el que se mantenga durante el proceso de creación de la misma. Tanto la opción *Normal* como la *Hard* no mantienen el tamaño de mallado definido por el usuario, sino que dan prioridad al tamaño de la estructura.

- ***Unstructured size transitions***. Este valor indica si las transiciones entre elementos con distintos tamaños se realizan rápido o despacio. Esta transición se produce desde la superficie de las estructuras hacia dentro del volumen y se denomina gradiente. Este valor está comprendido entre 0 y 1. Cuanto mayor sea el valor del gradiente, mayor será el cambio que se producirá en el volumen.

2.2. *Preprocesador NetGen.*

2.2.1. Descripción general.

NetGen es una herramienta de generación automática de malla tetraédrica en 3D. Fue desarrollado principalmente por Joachim Schöberl en la Universidad Johannes Kepler de Linz. *NetGen* es un software de código abierto bajo la *Licencia Pública General de GNU* disponible para Unix/Linux y Windows 98/NT [5].

En *NetGen*, la entrada de datos se realiza mediante un fichero con la información de la geometría del problema. Los dos formatos que acepta *NetGen* como entrada son un fichero con geometría constructiva de sólidos (*CSG*) o un fichero con la representación del contorno (*BREP*) en un formato *STL*. Además, *NetGen* contiene módulos para la optimización y refinamiento de malla jerárquica.

La principal diferencia que existe entre el fichero *STL* y el *CSG* es que el primero contiene la información del mallado de una estructura mientras que el segundo contiene la geometría de dicha estructura. Se ha comprobado que es mucho más eficiente para generar mallados de estructuras utilizar como entrada el formato de fichero *CSG*, ya que no es lo mismo realizar el mallado por primera vez de una estructura que realizar un mallado a partir de otro dado.

El formato de entrada *CSG* es muy útil para geometrías de diferentes tamaños, pero muy difícil de construir sin las herramientas adecuadas. Los archivos de tipo *CSG* contienen una serie de primitivas que representan un conjunto de estructuras y mediante las cuales *NetGen* es capaz de construirlas. Por lo tanto podemos decir que una primitiva es un conjunto de sentencias con una estructura determinada que definen unívocamente un tipo de estructura y la posición que ocupa en el espacio. Es importante destacar que, tal y como se ha comentado en la introducción, este fichero de geometría debe ser construído por el usuario manualmente, lo que se convierte en algo muy costoso cuando se desarollan problemas con una estructura muy compleja formados por un gran número de elementos.

Todo lo referente al contenido y la forma de este tipo de ficheros, junto con las primitivas que puede contener, se tratará en la sección 4.2.

A continuación, y a modo de ejemplo, se muestran una serie de estructuras creadas por la herramienta *NetGen* en las que se puede observar la complejidad que es capaz de alcanzar este generador de mallado.

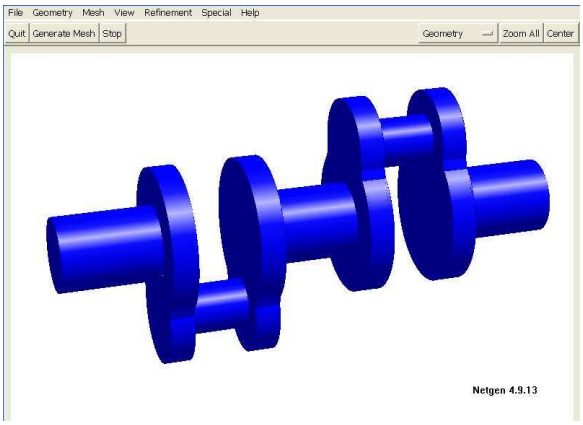


Figura 2.5: Representación en 3D de una estructura.

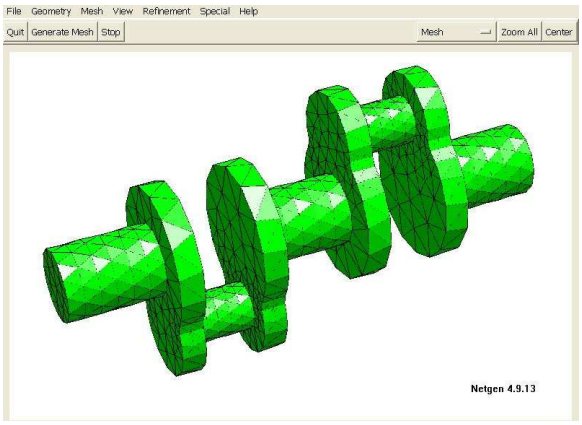


Figura 2.6: Mallado mallado de la estructura.

En la figura 2.5 podemos ver una estructura muy compleja representada en 3D a partir de un fichero que contiene su geometría y en la figura 2.6 se representa la malla que *NetGen* ha creado a partir de esos volumenes.

2.2.2. Generación de mallados mediante *NetGen.*

NetGen tiene la capacidad que tiene de crear mallados de diversas estructuras en 3D. Modificando una serie de parámetros en su menú *Meshing Options* podemos cambiar el tamaño de los tetraedros que forman la malla de la estructura, el orden de estos tetraedros, optimización de la malla, refinamiento, etc...

Se van a analizar las opciones de configuración que proporciona *NetGen* para modificar los mallados en función de las necesidades del usuario. En las figuras 2.7 y 2.8 se pueden observar las diferentes pestañas que contienen las opciones que existen para modificar los mallados generados con *NetGen.*

- ***Mesh granularity***. Indica el tamaño que tendrá la malla que va a ser generada por *NetGen.* Hay algunos tamaños que vienen definidos por defecto, como por ejemplo *very coarse, coarse, moderate, fine o very fine.* Además, la opción *User defined* permite que sea el usuario el que introduzca el tamaño de los elementos que quiere que tenga la malla.

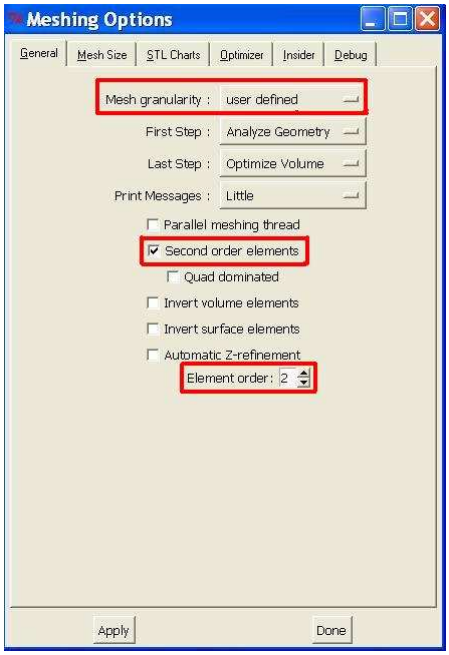


Figura 2.7: Menú de opciones generales.

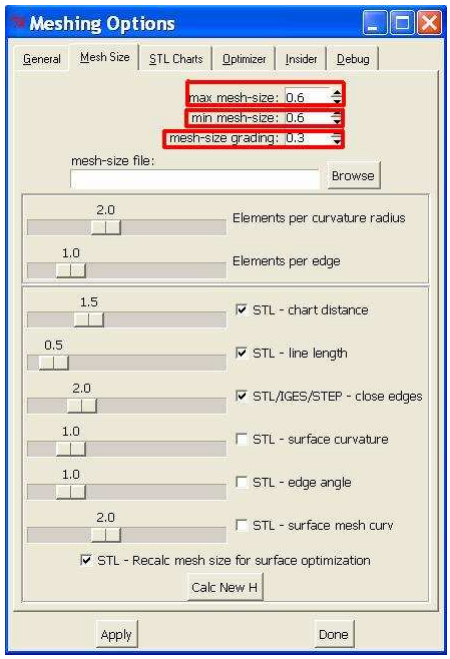


Figura 2.8: Menú para el tamaño de mallado.

- ***Second order elements***. Esta opción debe estar activa junto con *Element order* igual a dos para que los tetraedros que se generen sean cuadráticos.

- ***Max mesh-size***. Este valor indica el tamaño máximo que pueden tener los tetraedros. Será el valor que se tome como tamaño máximo de la malla si se selecciona la opción *User defined* en el menú.

- ***Min mesh-size***. Este valor indica el tamaño mínimo que pueden tener los tetraedros. Este valor se tomará como tamaño mínimo del a malla si se selecciona la opción *User defined* en el menú.

  Si utilizamos en estos dos parámetros el mismo valor conseguimos establecer un valor fijo para el tamaño de los tetraedros.

- ***Mesh-size grading***. Al igual que en *GiD*®, este valor indica si las transiciones entre elementos con distintos tamaños se realizan rápido o despacio. Esta transición se produce desde la superficie de las estructuras hacia dentro del volumen y se denomina gradiente. Este valor está comprendido entre 0 y 1. Cuanto mayor sea el valor del gradiente, mayor será el cambio que se producirá en el volumen.

Mediante la correcta configuración de estas opciones de mallado conseguimos que los dos generadores de mallado bajo estudio presenten comportamientos similares a la hora de generar

mallas. En el capítulo 5 se incluye un resumen con los parámetros utilizados.

2.3. Similitudes y diferencias entre *GiD* y *NetGen*.

- **Similitudes.**
  - Tanto *GiD*® como *NetGen* actúan de *preprocesador* y permiten discretizar dominios de problemas físicos.
  - Las dos herramientas generan mallados que se adaptan completamente a las estructuras iniciales y permiten exportar la información de puntos y conectividades a ficheros de texto.
  - *GiD*® y *NetGen* poseen una serie de opciones de configuración que permiten modificar las características de los generadores de mallado y obtener mallas diferentes en función del problema que se esté tratando.
  - Ambos permiten la asignación de condiciones de contorno a las superficies de las entidades geométricas.

- **Diferencias.**
  - La herramienta *GiD*® actúa también como *postprocesador*, mientras que *NetGen* no es capaz de analizar resultados.
  - *GiD*® es un generador de mallado que permite analizar estructuras en dos y tres dimensiones mientras que *NetGen* solamente es capaz de mallar problemas en tres dimensiones.
  - *GiD*® permite la realización de mallados mediante tetraedros, hexaedros o hexaedros en estructuras en 3D, sin embargo *NetGen* solamente utiliza tetraedros.
  - *GiD*® necesita licencia mientras que *NetGen* es software libre.
  - La interfaz gráfica de *GiD*® permite realizar diseños introduciendo gráficamente las entidades geométricas y sin embargo *NetGen* necesita un fichero de texto para poder representar estos volúmenes dentro de la interfaz.

Capítulo 3

Descripción de *GiDtoNet*

A lo largo de este capítulo se va a realizar una presentación del módulo *GiDtoNet* incluyendo los procedimientos de los que está compuesto para que el usuario pueda entender correctamente la implementación del mismo.

3.1. Introducción

Como ya se ha comentado en capítulos anteriores, la finalidad de este Proyecto Fin de Carrera es realizar una comparación entre las mallas proporcionadas por *GiD*® y *NetGen* sobre una misma estructura. Para no tener que crear manualmente el fichero geometría *CSG*, se ha desarrollado un módulo o *problem type* que permite generar las primitivas con la información de cada una de las entidades geométricas presentes en un proyecto *GiD*®. Este módulo se denomina *GiDtoNet*.

Este módulo añade en la interfaz de *GiD*® un nuevo menú con una serie de opciones que van a permitir que el usuario pueda construir el fichero *CSG* con la geometría de un problema sin tener que llevar a cabo su construcción manualmente y además disponer del mallado generado por *NetGen*.

GiDtoNet proporciona una ventana para llevar a cabo la creación las primitivas de las cuatro entidades geométricas soportadas por *NetGen*. Además implementa las tres operaciones Eulerianas que *NetGen* soportadas a la hora de crear alguna de estas entidades geométricas. *GiDtoNet* se encarga, mediante una serie de procedimientos Tcl, de extraer y almacenar la información de todas y cada una de las entidades geométricas representadas en *GiD*®. Además permite tener en todo momento actualizada la información de los elementos que estén en el proyecto en un momento determinado.

Una vez que se ha creado la estructura en la interfaz de *GiD*, el módulo *GiDtoNet* permite construir el fichero *CSG* con toda la información y con el formato adecuado para cargar esa mismo problema en *NetGen*. Una vez que el usuario dispone del fichero *CSG*, mediante una llamada remota a *NetGen*, podrá obtener el mallado de su problema realizado por *NetGen*. Esta malla se mostrará al usuario al pedir su generación, por lo tanto, una de las cualidades del módulo desarrollado en este Proyecto Fin de Carrera es que todo el proceso que se lleva a cabo hasta obtener el mallado final con *NetGen* es transparente al usuario.

Cuando el usuario dispone de la malla generada por *NetGen* puede utilizar una serie de herramientas que proporciona *GiD*® para medir las cualidades del mallado. Como se ha detallado en la sección 1.1, mediante estas herramientas se realizará un estudio posterior de las cualidades de los mallados obtenidos tras la utilización del módulo *GiDtoNet* y se compararán con los mallados que *GiD*® proporciona bajo unas características concretas.

Las cuatro entidades geométricas que soporta *NetGen* y que podrán ser generadas con el módulo *GiDtoNet* son: el hexaedro, la esfera, el cilindro y el tronco de cono.

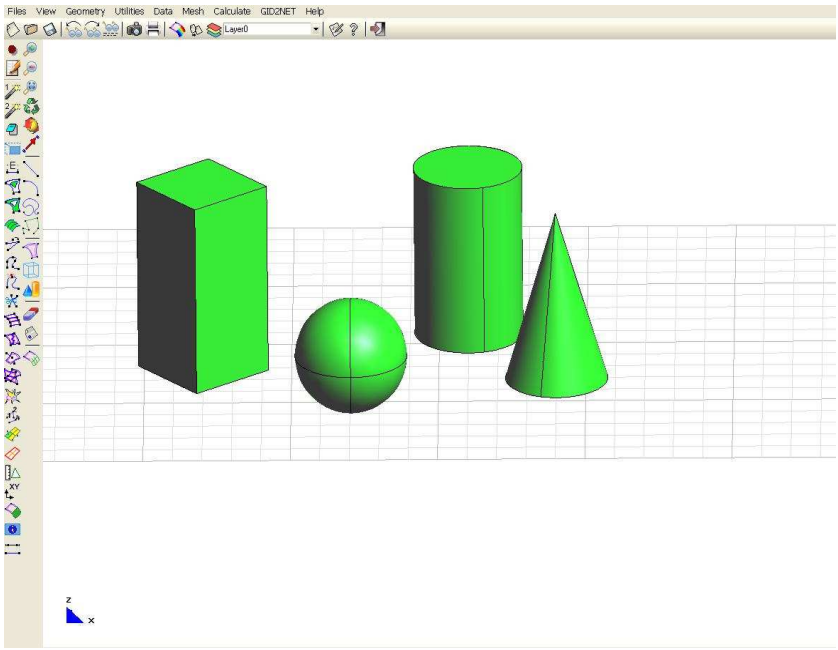


Figura 3.1: Entidades geométricas simples representadas con *GiD*.

En el caso del cono, la herramienta de mallado *NetGen* tiene una limitación, ya que sólo es capaz de representar y mallar troncos de cono y no el volumen completo. Por este motivo, el módulo *GiDtoNet* se encargará internamente de transformar los conos creados por el usuario en troncos de cono (en la misma posición y del mismo tamaño de base), eliminando el tercio superior de los mismos, para que *NetGen* pueda interpretar correctamente esta entidad geométrica. Se ha demostrado mediante una serie de pruebas que eliminando el tercio superior del cono se conseguía un mallado completo de la estructura por parte de *NetGen*, mientras que para otro casos no se conseguía realizar dicho mallado.

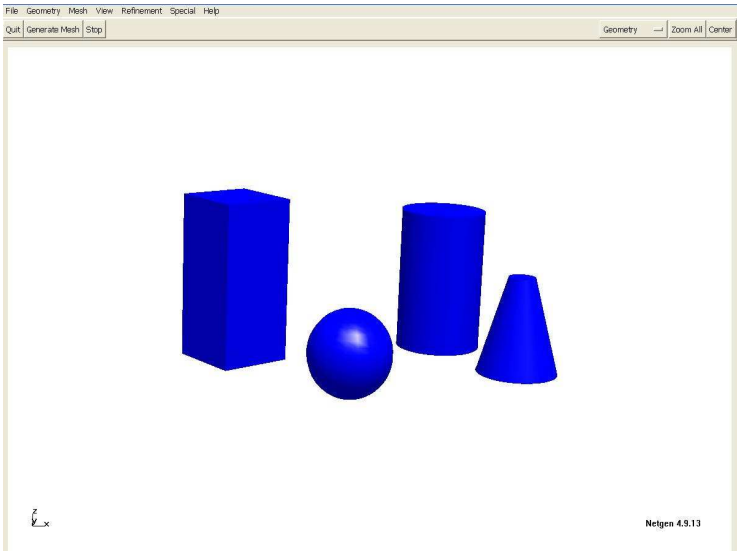


Figura 3.2: Entidades geométricas simples representadas con *NetGen*.

Al hexaedro, cono, cilindro y esfera las llamaremos a partir de ahora entidades geométricas simples y a aquellas que proceden de una operación booleana compuestas. La única limitación que tiene el módulo *GiDtoNet* es que dichas operaciones booleanas sólo se podrán realizar entre dos entidades geométricas simples, dejando para futuras mejoras el resto de posibilidades. Estas entidades geométricas pueden observarse en la figura 3.1.

A continuación se enumeran los pasos que el usuario debe seguir para aprovechar al máximo las capacidades del módulo *GiDtoNet*.

- **Paso 1: Inicialización de *GiDtoNet*.**

  El primer paso que debe dar el usuario es la carga del módulo en *GiD*®. Al realizar esta acción, *GiD*® se configura y permite utilizar todas las herramientas implementadas por el módulo *GiDtoNet*.

- **Paso 2: Creación de las entidades geométricas simples.**

  Para que el usuario tenga acceso a crear las entidades geométricas, se le ha facilitado una ventana en la que se pueden encontrar los cuatro tipos de volúmenes que se tienen disponibles. Las funciones de esta ventana y su creación se explicarán más adelante en la sección 3.3.

- **Paso 3: Modificación del proyecto.**

  Aunque la modificación del proyecto se puede llevar a cabo en cualquier momento, se ha decidido incluir un paso de la utilización del módulo *GiDtoNet* para explicarle al usuario el proceso de actualización de la información del proyecto. Éste es uno de los momentos más importantes del uso de *GiDtoNet*, ya que gracias al almacenamiento de estos datos podemos llevar a cabo más tarde el traspaso de información de la geometría del problema a *NetGen* mediante el fichero *CSG*. Se considera que el proyecto se ha modificado si se elimina alguna entidad geométricas o si se añade alguna nueva a un proyecto ya creado. Esta forma de actualizar continuamente la situación del proyecto le confiere al módulo *GiDtoNet* una gran capacidad de almacenamiento y procesado de la información con la que está trabajando. Todo este desarrollo se encuentra descrito en la sección 3.4.

- **Paso 4: Asignación de las condiciones de contorno.**

  Las condiciones de contorno se asignarán sobre las superficies de las entidades geométricas simples, antes siempre de realizar cualquier operación booleana, ya que una vez realizada no se contempla la posibilidad de asignar condiciones de contorno. Por lo tanto, las entidades geométricas compuestas heredarán las condiciones de contorno que tuvieran las entidades simples de las que proceden.

- **Paso 5: Creación de entidades geométricas compuestas.**

  Si se desea realizar alguna operación booleana con las entidades geométricas simples se puede realizar ahora o en cualquier otro momento siempre y cuando respetemos el paso anterior y asignemos las condiciones de contorno antes de realizar dichas operaciones. Para elegir la operación booleana que se va a realizar no hay más que mostrar de nuevo la ventana y elegir el botón correspondiente.

- **Paso 6: Guardar el proyecto y crear el fichero *CSG*.**

  Antes de mallar el proyecto que el usuario está desarrollando, debe ser guardado, ya que si se intenta llevar a cabo la creación del fichero *CSG* antes de almacenar la información del proyecto, se recibirá un mensaje de error comunicándonos que lo guardemos. El fomato del fichero *CSG*, su contenido y su finalidad se explicarán detalladamente en el capítulo 4.

- **Paso 7: Mallar las entidades geométricas.**

  Una vez generado el fichero *CSG* ya estamos preparados para mallar las estructuras presentes en el proyecto. Al dar la orden a *GiD*® el módulo *GiDtoNet* llamará internamente a *NetGen* con el fichero *CSG* creado en el paso anterior e importará la malla desde *NetGen*. De esta forma, el usuario únicamente verá que se ha creado un nuevo fichero coon la información del mallado final de su proyecto. Una vez obtenidos los mallados se puede realizar un estudio de las calidades del mallado resultante mediante las herramientas que proporciona *GiD*® en el menú *Mesh quality*.

Ya estamos preparados para presentar el diagrama de flujo del módulo, en la figura 3.3. En éste podemos encontrar los pasos que se siguen desde el momento en el que cargamos el módulo hasta el que recuperamos el mallado dado con la herramienta *NetGen*.

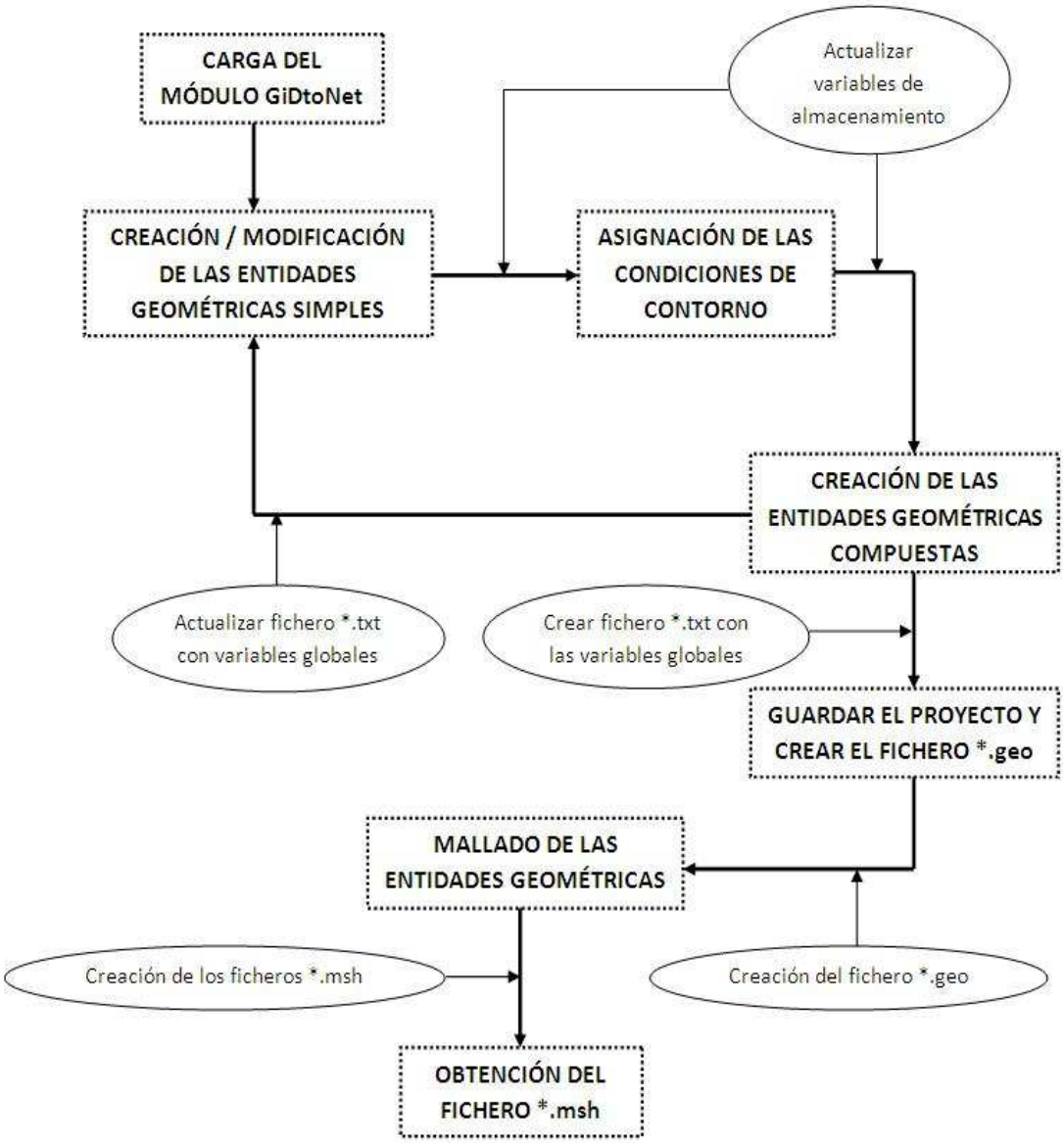


Figura 3.3: Diagrama de flujo para la correcta utilización del módulo *GiDtoNet*.

3.2. Paso 1: Inicialización del módulo *GiDtoNet*.

Si el usuario quiere utilizar el módulo *GiDtoNet* lo primero que tiene que hacer es cargarlo en *GiD*®. Para ello, hay que seleccionar en el menú **data -> problem type -> *GiDtoNet***. Una vez realizado este simple paso comenzará la carga de *GiDtoNet* y la interfaz que añade el módulo estará disponible para que el usuario las utilice a su gusto.

Al cargar el módulo, internamente *GiD*® ejecuta un procedimiento que se conoce como *InitGiDProject* que suele contener todas las acciones para cargar menús, barras de herramientas y en general todos aquellos elementos que caracterizan la apariencia del módulo. En el pseudocódigo 3.1 se pueden observar una serie de líneas que indican las principales acciones de este procedimiento Tcl.

A lo largo de este Proyecto Fin de Carrera siempre se que se realice el análisis de cualquiera de los procedimientos que están involucrados en el módulo *GiDtoNet*, se adjuntará el pseudocódigo del mismo, incluyendo únicamente las líneas que se han considerado más importantes, para que el usuario pueda tener un conocimiento general del funcionamiento interno del módulo. Si se desea más información, se podrá consultar el código completo del módulo en el apéndice C.

```
1 proc InitGIDProject { dir } {
2
3     #Comando para insertar la presentacion
4     GidUtils::Splash [file join directory Images "image"]
5
6     #Comando para insertar menus
7     GiDMenu::Create [= "Name"] "PRE"
8
9     #Comando para insertar opciones
10    GiDMenu::InsertOption [= "Name"] "Option" \
11    0 PRE [= "Command"]
12    .
13    .
14    .
15
16    GiDMenu::InsertOption [= "Name"] "Option"  \
17    3 PRE [= "Command"]
18
19    #Comando para actualizar los menus
20    GiDMenu::UpdateMenus
21
22    #Comando para eventos
23    bind .gid <KeyPress-Escape> [= "Command"]
24    bind .gid <KeyRelease-Escape> [= "Command"]
25 }
```

Pseudocódigo 3.1: Código del procedimiento *InitGiDProject*.

- Línea 4. Comando que permite cargar la imagen de presentación del módulo.
- Línea 7. Este comando permite crear un nuevo menú en *GiD*®. El string *Name* indica el nombre que se le dará al menú y la opción *PRE* indica que el menú se crea en la parte de preproceso.
- Líneas 10-16. Comandos que permiten crear las distintas opciones del menú anterior. El string *Name* identifica el menú en el que se quieren añadir estas opciones, *Option* es el string que indica el nombre de la opción a insertar y *Command* identifica el comando a ejecutar cuando se selecciona la opción.
- Línea 20. Este comando carga el nuevo menú en *GiD*®.
- Líneas 23-24. Estos comandos Tcl-Tk premiten identificar un evento que el usuario realiza mediante teclado (en este caso pulsar la tecla escape) con un procedimiento.

En la figura 3.4 se puede observar la interfaz gráfica del módulo *GiDtoNet* y los menús que se han añadido para acceder a las nuevas funcionalidades aportadas por dicho módulo.

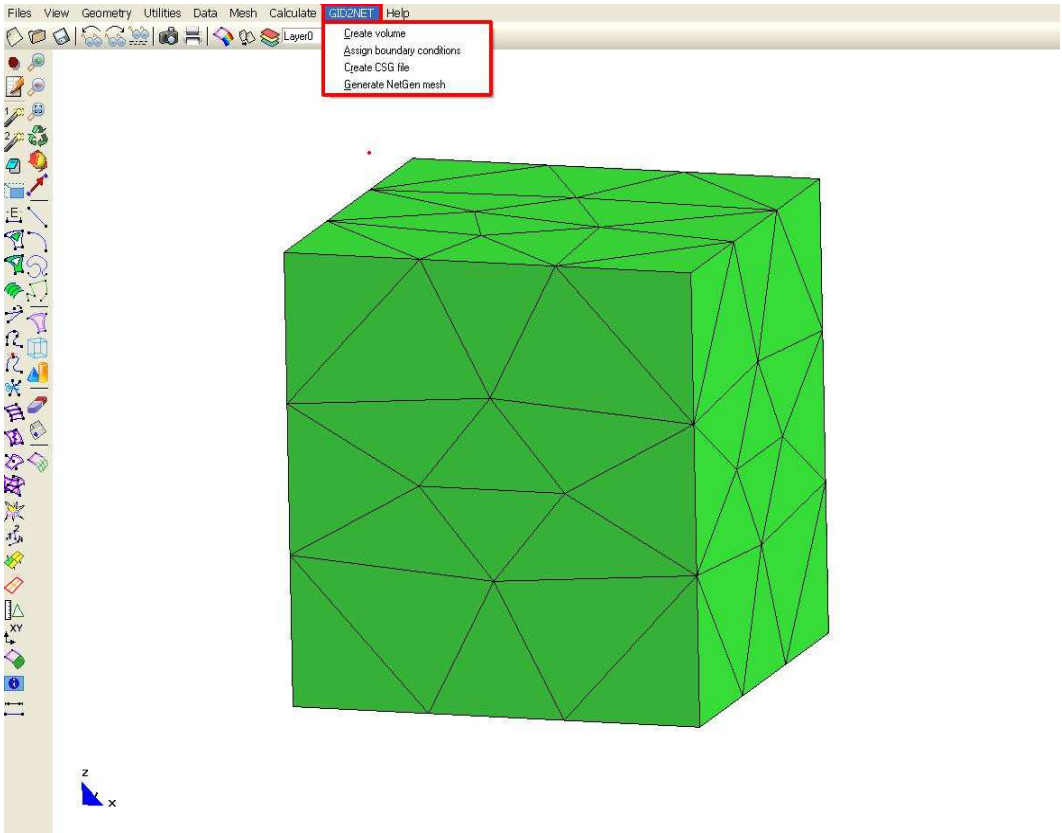


Figura 3.4: Interfaz del módulo *GiDtoNet*.

3.3. Paso 2: Creación de entidades geométricas.

Una vez cargado el módulo *GiDtoNet*, el usuario puede empezar a dar forma a su proyecto. Para ello debe comenzar con la creación de entidades geométricas simples en función de sus necesidades.

3.3.1. Presentación de la ventana de selección.

En el menú **GiD2Net -> Crear volumen** se accederá a la ventana que vemos en la figura 3.5. En ella, se podrá elegir entre las cuatro entidades geométricas de las que dispone el módulo *GiDtoNet* para desarrollar el proyecto.

Esta ventana contiene además tres botones para realizar posteriormente las operaciones Eularianas que están permitidas en el módulo. Estas operaciones se detallarán en la sección 3.6.

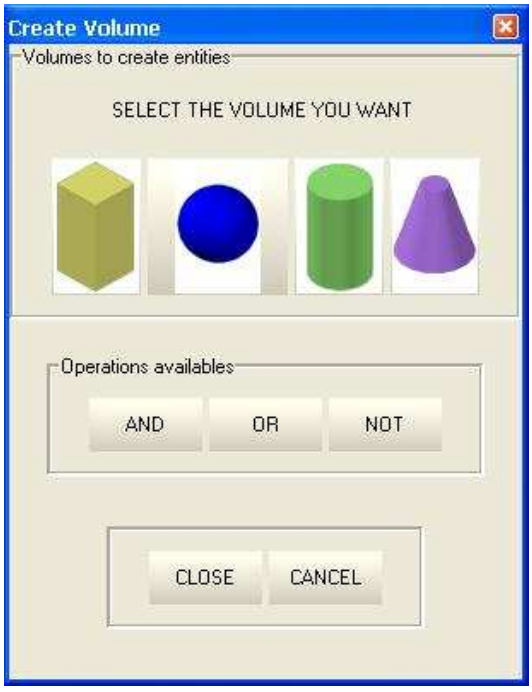


Figura 3.5: Ventana de selección de entidades geométricas simples.

El procedimiento que se encarga de mostrar esta ventana se denomina *Create_volume*. Como se puede observar en su pseudocódigo (véase pseudocódigo 3.2) crea los diferentes menús y los relaciona con una serie de comandos para que se ejecuten unos procedimientos determinados al seleccionar cada uno de ellos.

```
1 proc Create_volume {} {
2
3   #Crear las variables para todas las imagenes de los botones
4   set "var" [image create photo -file [file join directory Images "image"]]
5   .
6   .
7   set "var" [image create photo -file [file join directory Images "image"]]
8
9   #Inicializa la ventana
10  InitWindow [= "Window"]  [= "Text"] c_volum "" "" 1
11
12  #Crear y visualizar las etiquetas para los marcos
13  labelframe [= "FrameName"] -text [= "Text"]
14  .
15  .
16  labelframe [= "FrameName"] -text [= "Text"]
17  pack [= "FrameName"]
18
19  #Crear y visualizar las etiquetas
20  label [= "LabelName"] -text [= "Text"]
21  pack [= "Name"] -anchor "Position"
22
23  #Crear y visualizar los botones
24  button [= "ButtonName"] -image "image" -height "height" \
25    -width "width" -command [= "Command"]
26  .
27  .
28  .
29  button [= "ButtonName"] -image "image" -height "height" \
30    -width "width" -command [= "Command"]
31  pack [= "ButtonName"]  -side "side" -anchor "anchor"
32 }
```

Pseudocódigo 3.2: Código del procedimiento *Create_volume*.

- Líneas 4-7. Comandos que permiten crear unas variables que están relacionadas con las imágenes que poseen cada uno de los botones. El string *var* es el nombre que tomará dicha variable e *image* es el fichero *.gif que contiene la imagen.

- Línea 10. Se crea la ventana que contendrá los elementos para la creación de volúmnenes y operaciones. El string *Window* indica el nombre que recibirá la ventana y *Text* el texto que tendrá al visualizarla.

- Líneas 13-16. Comandos que permiten crear los marcos que contendrá la ventana. El string *FrameName* indica el nombre que recibe en codificación el marco y *Text* el texto con el que se visualizará.

- Línea 17. Comando que permite visualizar dentro de la ventana el elemento representado por el string *FrameName*.

- Línea 20. Comando que permite crear una etiqueta para la ventana. El string *LabelName* indica el nombre que recibe en codificación dicha etiqueta y *Text* el texto con el que se mostrará.

- Línea 21. Comando que permite visualizar dentro de la ventana el elemento representado por el string *LabelName* en la posición indicada por *Position*.

- Líneas 24-30. Comandos que permiten crear los botones que contendrá la ventana. El string *ButtonName* indica el nombre que recibe en codificación el botón, *image* representa la foto que contendrá, *height* y *width* son las dimensiones del botón y *Command* el comando que se ejecutará al pulsarlo.

- Línea 31. Comando que permite visualizar dentro de la ventana el elemento representado por el string *ButtonName* en la posición indicada por los strings *side* y *anchor*.

Al crear una entidad geométrica, entra en funcionamiento el procedimiento llamado *Pulsa_boton* y que es el encargado de crear el volumen dentro del proyecto. Este procedimiento necesita un parámetro que indicará el tipo de volumen que se va a crear. El código de *Pulsa_boton* viene recogido en el pseudocódigo 3.3.

```
1 proc Pulsa_boton {tipo_volumen} {
2
3   Button_scape
4
5   #Indicar el tipo de volumen actual
6   set actual_type [...]
7
8   #Crear el volumen
9   GiD_Process Mescape Geometry Create Object "Type"
10 }
```

Pseudocódigo 3.3: Código del procedimiento *Pulsa_boton*.

- Línea 3. Llamada a un procedimiento permite cambiar de contexto almacenando todos los cambios que se hayan podido producir. Se describirá más adelante.

- Línea 6. Sentencia que almacena el valor del tipo de volumen que se está creando para futuras operaciones.

- Línea 9. Este comando permite crear un volumen del tipo indicado por el string *Type* dentro del proyecto.

3.3.2. Almacenamiento de la información de los volúmenes creados.

Una vez creada la entidad geométrica a partir de cada uno de los botones de la figura 3.5, el módulo se encargará internamente de almacenar la información de todos y cada uno de ellos. Este procedimiento es totalmente transparente para el usuario.

En función del volumen que el usuario desee crear se han desarrollado cuatro procedimientos diferentes. Éstos extraen la información de las entidades geométricas de una forma distinta en función de su tipo. Esta información se utilizará más tarde para construir el fichero *CSG*, ver sección 3.7.

Los pseudocódigos de estos procedimientos son el 3.4, 3.5, 3.6 y 3.7. Como se puede observar, todos ellos necesitan un parámetro de entrada indicando el identificador que tiene el volumen del que queremos extraer la información.

Hay que destacar que para identificar los tipos de volúmenes presentados en este Proyecto Fin de Carrera, se define una condición *GiD* que permite relacionar cada tipo de volumen con un entero de manera unívoca. Dicha condición *GiD* nos va a permitir que a lo largo de la vida del proyecto podamos identificar qué tipo de entidades geométricas se están manejando. Los valores asignados en función del tipo de entidad son enteros entre el cero y el tres. La asignación de esta condición por parte del módulo se realizará en el momento de la creación del volumen. Mediante la lógica implementada en el módulo, a la hora de acceder a la información de ese volumen se puede tener disponible su condición *GiD*, lo que permitirá identificar claramente qué tipo de elemento se está procesando. Tanto la asignación como la consulta de estas condiciones de *GiD* se llevan a camo mediante unos comandos destinados a ello.

Así, por ejemplo, si creamos una esfera con identificador de volumen *id*, su condición *GiD* se inicializará a 1. Cada vez que se acceda al identificador de volumen *id* y se extraiga la condición *GiD* asociada a este identificador, se obtendrá un 1, por lo que sabemos que dicho volumen es una esfera. Estos tipos de condiciones se detallan en la sección 4.3.

- Hexaedro.

  El primero de los procedimientos que vamos a analizar es el encargado de extraer la información del hexaedro, cuyo pseudocódigo es el 3.4. Esta entidad geométrica es la que necesita un mayor número de parámetros ya que está formada por seis planos cuya intersección genera el volumen deseado.

```
1 proc Add_prism {id_volumen} {
2
3   #Posicion que ocupara el volumen dentro de las listas
4   set indice [...]
5   #Asignar condicion GiD
6   GiD_AssignData condition Type Volumes "Valor" "ID"
7   set information_volumen [GiD_Info list_entities Volumes "ID" -sublist]
8   #Extraer identificadores de superficies
9   set s1 [lindex "information_volumen" "position"]
10  set s6 [lindex "information_volumen" "position"]
11  #Extraer informacion de superficies
12  set inf_s1 [GiD_Info list_entities surfaces "ID_surface" -sublist]
13  set inf_s6 [GiD_Info list_entities surfaces "ID_surface" -sublist]
14  #Obtencion de la condicion de contorno
15  set bc [GiD_Info Conditions Boundary geometry "ID_surface"]
16  #Obtener de los centros de cada uno de los planos
17  set ::GID2NET::LIST_INFO(TYPE,"Indice") "Type"
18  #Almacenar informacion
19  set ::GID2NET::LIST_INFO("Plane","Indice") \
20  "plane (" Center";"Normal") -bc = "BC""
21  set ::GID2NET::LIST_FILE("Indice") [= "Text"]
22 }
```

Pseudocódigo 3.4: Procedimiento *Add_prism*.

- Línea 4. La variable *indice* indica la posición en la que se almacenará la información del volumen que estamos procesando dentro de las listas.
- Línea 6. Este comando permite asignar al volumen la condición *GiD*® en función del tipo de entidad que corresponda. El string *Type* representa el valor de la condición *GiD*® e *ID* es el identificador del volumen.
- Línea 7. Mediante este comando se extrae en una lista la información del volumen que estamos tratando. El string *ID* indica el identificador de este volumen.
- Líneas 9-10. Extraen los identificadores de las superficies del hexaedro recorriendo la lista que contiene la información del volumen. El string *position* indica en qué lugar está la información que queremos obtener.
- Líneas 12-13. Este comando permite obtener la información que poseen las superficies que forman el hexaedro. El string *ID_surface* indica el identificador de cada una de ellas.
- Línea 15. Este comando obtiene información sobre las condiciones de contorno que poseen cada una de las superficies del hexaedro. Es el string *ID_surface* el que representa el identificador de estas superficies.
- Línea 17. Esta sentencia permite almacenar el tipo de volumen que se está tratando en una lista dedicada a ello. El string *Indice* indica la posición en la que se almacena la información y *Type* es el valor del tipo.
- Líneas 19-21. Estas sentencias permiten almacenar la información del volumen que se ha obtenido a lo largo del procedimiento. El string *Plane* indica el plano que se está almacenando, *Indice* es la posición que ocupa el volumen dentro de la lista. *Center*, *Normal* y *BC* son los strings que indican las características de los planos y *Text* es la sentencia que se imprimirá en el fichero *CSG*, que se explicará detalladamente en la sección 3.7.

- Esfera.

  El siguiente pseudocódigo del que disponemos es el del procedimiento *Add_sphere* (véase pseudocódigo 3.5) que difiere del anterior en que los datos que se deben extraer para construir su primitiva son diferentes. Además como se ha comentado anteriormente, se le asignará a este tipo de volumen un identificador que nos permitirá saber qué estamos tratando.

  - Línea 4. La variable *indice* indica la posición en la que se almacenará la información del volumen que estamos procesando en las listas.
  - Línea 7. Este comando permite asignar al volumen la condición *GiD*® en función del tipo de entidad que corresponda. El string *Type* representa el valor de la condición *GiD*® e *ID* es el identificador del volumen.
  - Línea 8. Mediante este comando se extrae en una lista la información del volumen que estamos tratando. El string *ID* indica el identificador de este volumen.

- Líneas 11-13. Extraen los identificadores de las superficies de la esfera recorriendo la lista que contiene la información del volumen. El string *position* indica en qué lugar está la información que queremos obtener.

```
1 proc Add_sphere {id_volumen} {
2
3   #Posicion que ocupara el volumen dentro de las listas
4   set indice [...]
5
6   #Asignar condicion GiD
7   GiD_AssignData condition Type Volumes "Valor" "ID"
8   set information_volumen [GiD_Info list_entities Volumes "ID" -sublist]
9
10  #Extraer identificadores de superficies
11  set s1 [lindex "information_volumen" "position"]
12  ...
13  set s4 [lindex "information_volumen" "position"]
14
15  #Obtencion del centro de la esfera
16
17  #Extraer informacion de superficies
18  set inf_s1 [GiD_Info list_entities surfaces "ID_surface" -sublist]
19  ...
20
21  #Obtener de la informacion de lineas
22
23  #Extraer informacion de puntos
24  set info_point1 [GiD_Info list_entities points "ID_point" -sublist]
25
26  #Obtener de la condicion de contorno
27  set dato_bc [GiD_Info Conditions Boundary geometry "ID_surface"]
28
29  set ::GID2NET::LIST_INFO(TYPE,"Indice") "Type"
30
31  #Almacenar informacion de volumen
32  set ::GID2NET::LIST_FILE("Indice") \
33  "sphere ("Center";"Radio") -bc = "BC""
34 }
```

Pseudocódigo 3.5: Procedimiento *Add_sphere*.

- Línea 18. Este comando permite obtener la información de cada una de las superficies que forman la esfera. El string *ID_surface* indica el identificador de cada una de ellas.
- Línea 24. Este comando obtiene información de cada uno de los puntos que son necesarios para crear la primitiva de la esfera. Es el string *ID_point* el que representa el identificador de cada uno de ellos.
- Línea 27. Este comando obtiene información sobre las condiciones de contorno que poseen cada una de las superficies de la esfera. Es el string *ID_surface* el que representa el identificador de estas superficies.
- Línea 29. Esta sentencia permite almacenar el tipo de volumen que se está tratando en una lista dedicada a ello. El string *Indice* indica la posición en la que se almacena la información y *Type* es el valor del tipo.
- Líneas 32-33. Estas sentencias permiten almacenar la información del volumen que se ha obtenido a lo largo del procedimiento. El string *Indice* es la posición que ocupa el volumen dentro de la lista. *Center*, *Radio* y *BC* son los strings que indican las características de la esfera.

- Cilindro.

A continuación, el pseudocódigo 3.6 muestra el procedimiento *Add_cilinder*. En este caso podemos encontrar más semejanzas con el primer pseudocódigo que hemos visto, el que se corresponde con *Add_prism*.

```
1 proc Add_cylinder {id_volumen} {
2
3   #Posicion que ocupara el volumen dentro de las listas
4   set indice [...]
5
6   #Asignar condicion GiD
7   GiD_AssignData condition Type Volumes "Valor" "ID"
8
9   #Obtener informacion de volumen
10  set information_volumen [GiD_Info list_entities Volumes "ID" -sublist]
11
12  #Obtener informacion de superficies
13  set inf_s1 [GiD_Info list_entities surfaces "ID_surface" -sublist]
14
15  #Obtener un punto y la normal de las superficies
16
17  #Extraer informacion de lineas
18  set inf_l1 [GiD_Info list_entities lines "ID_line" -sublist]
19
20  #Extraer informacion de puntos
21  set info_point1 [GiD_Info list_entities points "ID_point" -sublist]
22
23  #Obtener de la condicion de contorno
24  set dato_bc [GiD_Info Conditions Boundary geometry "ID_surface"]
25
26  set ::GID2NET::LIST_INFO(TYPE,"Indice") "Type"
27
28  #Almacenar informacion de volumen
29  set ::GID2NET::LIST_INFO("Plane","Indice") \
30  "plane (""Center"";""Normal"") -bc = ""BC"""
31  set ::GID2NET::LIST_FILE("Indice") \
32  "cylinder (""Center"";""Center"";""Radio"") -bc = ""BC"""
33 }
```

Pseudocódigo 3.6: Procedimiento *Add_cylinder*.

- Línea 4. La variable *indice* indica la posición en la que se almacenará la información del volumen que estamos procesando dentro de las listas.
- Línea 7. Este comando permite asignar al volumen la condición *GiD*® en función del tipo de entidad que corresponda. El string *Type* representa el valor de la condición *GiD*® e *ID* es el identificador del volumen.
- Línea 10. Mediante este comando se extrae en una lista la información del volumen que estamos tratando. El string *ID* indica el identificador de este volumen.
- Líneas 13. Este comando permite obtener la información de cada una de las superficies que forman el cilindro. El string *ID_surface* indica el identificador de cada una de ellas.
- Líneas 18. Este comando permite obtener la información de cada una de las líneas que son susceptibles de almacenar información importante sobre el volumen que estamos tratando. El string *ID_line* representa el identificador de cada una de estas líneas.

- Línea 21. Este comando obtiene la información de cada uno de los puntos que son importantes en el volumen. El string *ID_point* es el que representa el identificador de estos puntos.
- Línea 24. Este comando obtiene información sobre las condiciones de contorno que poseen las superficies del cilindro. Es el string *ID_surface* el que representa el identificador de estas superficies.
- Línea 26. Esta sentencia permite almacenar el tipo de volumen que se está tratando en una lista dedicada a ello. El string *Indice* indica la posición en la que se almacena la información y *Type* es el valor del tipo.
- Líneas 29-32. Estas sentencias permiten almacenar la información del volumen que se ha obtenido a lo largo del procedimiento. El string *Plane* indica el plano que se está almacenando, *Indice* es la posición que ocupa el volumen dentro de la lista y *Center*, *Normal* y *BC* son los strings que indican las características de los planos que se están almacenando. *Center*, *Radio* y *BC* son los datos importantes de la cara lateral del cilindro.

- Cono.

El último pseudocódigo que se va a presentar es el del procedimiento *Add_cone* y se puede encontrar en el pseudocódigo 3.7. Éste sigue la estructura de los procedimientos anteriores cambiando alguno de los datos que son necesarios para crear su primitiva.

- Línea 4. La variable *indice* indica la posición en la que se almacenará la información del volumen que estamos procesando en las listas.
- Línea 7. Este comando permite asignar al volumen la condición *GiD*® en función del tipo de entidad que corresponda. El string *Type* representa el valor de la condición *GiD*® e *ID* es el identificador del volumen.
- Línea 10. Mediante este comando se extrae en una lista la información del volumen que estamos tratando. El string *ID* indica el identificador de este volumen.
- Líneas 13. Este comando permite obtener la información de cada una de las superficies que forman el hexaedro. El string *ID_surface* indica el identificador de cada una de ellas.
- Línea 18. Este comando permite obtener la información de cada una de las líneas que son susceptibles de almacenar información importante sobre el volumen que estamos tratando. El string *ID_line* representa el identificador de cada una de estas líneas.
- Línea 22. Este comando obtiene la información de cada uno de los puntos que son importantes en el volumen. El string *ID_point* es el que representa el identificador de estos puntos.

```
1 proc Add_cone {id_volumen} {
2
3   #Posicion que ocupara el volumen dentro de las listas
4   set indice [...]
5
6   #Asignar condicion GiD
7   GiD_AssignData condition Type Volumes "Valor" "ID"
8
9   #Obtener informacion de volumen
10  set information_volumen [GiD_Info list_entities Volumes "ID" -sublist]
11
12  #Obtener informacion de superficies
13  set inf_s1 [GiD_Info list_entities surfaces "ID_surface" -sublist]
14
15  #Obtener un punto y la normal de las superficies
16
17  #Extraer informacion de lineas
18  set inf_l1 [GiD_Info list_entities lines "ID_line" -sublist]
19  ...
20
21  #Extraer informacion de puntos
22  set info_point1 [GiD_Info list_entities points "ID_point" -sublist]
23
24  #Obtener de la condicion de contorno
25  set dato_bc [GiD_Info Conditions Boundary geometry "ID_surface"]
26
27  set ::GID2NET::LIST_INFO(TYPE,"Indice") "Type"
28
29    #Almacenar informacion de volumen
30  set ::GID2NET::LIST_INFO("Plane","Indice") \
31  "plane ("Point";"Normal") -bc = "BC""
32  set ::GID2NET::LIST_FILE("Indice") \
33  "cone ("Center";"Radio";"Center";"Radio") -bc = "BC""
34 }
```

Pseudocódigo 3.7: Procedimiento *Add_cone*.

- Línea 25. Este comando obtiene información sobre las condiciones de contorno que poseen las superficies del cono. Es el string *ID_surface* el que representa el identificador de estas superficies.
- Línea 27. Esta sentencia permite almacenar el tipo de volumen que se está tratando en una lista dedicada a ello. El string *Indice* indica la posición en la que se almacena la información y *Type* es el valor del tipo.
- Líneas 30-33. Estas sentencias permiten almacenar la información del volumen que se ha obtenido a lo largo del procedimiento. El string *Plane* indica el plano que se está almacenanado, *Indice* es la posición que ocupa el volumen dentro de la lista y *Center*, *Normal* y *BC* son los strings que indican las características de los planos que se están almacenando. *Center*, *Radio* y *BC* son los datos importantes de la cara lateral del cilindro.

3.4. Paso 3: Modificación del proyecto.

En esta sección se va a explicar la que se considera parte fundamental del módulo *GiDtoNet*. Cuando hablamos de modificar el proyecto, nos referimos a añadir nuevas entidades geométricas o eliminar algunas de las ya existentes. Es decir, todo lo que suponga un cambio en las bases de datos que contienen la información de las entidades geométricas presentes en el proyecto.

Si el usuario quiere eliminar un volúmen cualquiera, no tiene más que, con la herramienta que le proporciona *GiD*®, seleccionar qué volumen desea eliminar y a continuación pulsar la tecla escape. Este simple gesto implica que se ejecute automáticamente un procedimiento, denominado *Button_escape* que permite almacenar de nuevo toda la información de las entidades geométricas que continúan formando parte del proyecto, eliminando de las bases de datos todo lo referente al volumen selecciónado para su eliminación.

A continuación, en el pseudocódigo 3.8 se muestra el procedimiento *Button_scape*, que va a permitir que el usuario comprenda mejor la funcionalidad de este procedimiento.

```
1  proc Button_scape {} {
2
3   #valor de los volumenes actuales
4   set actual_volumes [GiD_Info Geometry NumVolumes]
5
6   if {"actual_volumes" < "volumenes_almacenados"} {
7     All_volumes
8   }
9
10  #Hay una nueva operacion AND NOT
11  if {"op_andnot" == 1} {
12    All_volumes
13  }
14
15  #Si se han asignado condiciones de contorno
16  if {"bc" == 1} {
17    All_volumes
18  }
19
20  #Aumenta el numero de volumenes
21  if {"actual_volumes" > "volumenes_almacenados"} {
22    #numero de nuevos volumenes
23    set num_new_vol [expr "actual_volumes" - "volumenes_almacenados"]
24    set "volumenes_almacenados" [expr "volumenes_almacenados" + "num_new_vol"]
25    #crear los volumnes
26    Add_volume "num_new_vol"
27  }
28 }
```

Pseudocódigo 3.8: Procedimiento *Add_cone*.

- Línea 4. Comando que permite obtener el valor actual de volúmenes estes en el proyecto.
- Líneas 6-8. Si se ha eliminado algún volumen vuelve a extraer la información de todos los que permanecen en el proyecto para actualizar las bases de datos mediante el procedimiento Add_volumes.

- Líneas 11-13. Si se ha creado una nueva operación *and_not* tenemos que volver a almacenar la información de los volúmenes, ya que alguno de ellos se ha eliminado para dar paso a la entidad geométrica compuesta.

- Líneas 16-18. Mediante estas líneas se comprueba si se han asignado condiciones de contorno a alguna de las entidades geométricas del proyecto. En ese caso, se debe extraer dicha información y actualizar las listas qeu contienen la información de los volúmenes.

- Líneas 21-27. Si se ha creado algún volumen nuevo, lo primero que tenemos que hacer es saber cuántos se han creado, indicado por el string *num_new_vol*. Una vez que lo sabemos los creamos mediante el método *Add_volumen*.

3.5. Paso 4: Asignación de condiciones de contorno.

Como ya se ha comentado en la sección 3.1, las condiciones de contorno deben asignarse siempre antes de realizar operaciones entre entidades geométricas simples, ya que es la única forma de mantenerlas tras dichas operaciones.

De nuevo contamos con una ventana, véase figura 3.6, que permitirá que el usuario seleccione las condiciones de contorno que crea más convenientes sobre las superficies de las entidades geométricas simples que ha creado. Para ello debe ir al menú **GiD®2NET -> Assign boundary conditions**.

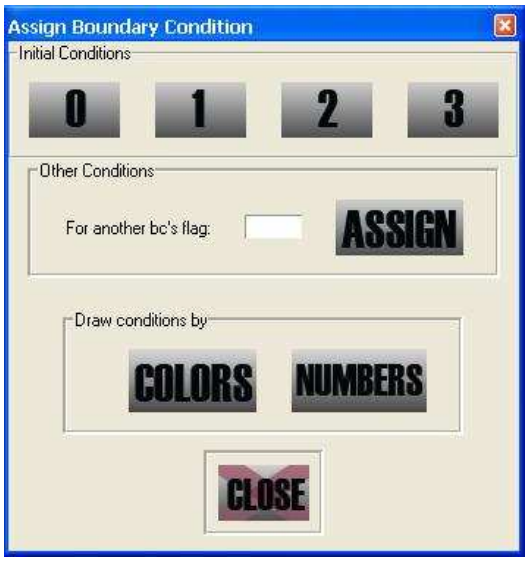


Figura 3.6: Ventana para la asignación de las condiciones de contorno.

El procedimiento *Window_bc* es el encargado de crear la ventana para la asignación de las condiciones de contorno. En el pseudocódigo 3.9 podemos encontrar la descripción del procedimiento para la obtención de dicha ventana.

```
1 proc Window_bc {} {
2
3   set [= "var"] [image create photo -file [file join "directory" Images [= "Image"]]]
4   #Crear ventana
5   InitWindow [= "Window"]  [= "Text"] c_volum "" "" 20
6   #Crear y visualizar los marcos
7   labelframe [= FrameName] -text [= "Text"]
8   pack [= FrameName] -side "side"
9   #Crear y visualizar los botones
10  button [= "ButtonName"] -image [= "Image"] -height "height" \
11    -width "width" -command [= "Command"]
12  pack [= "ButtonName"] -side "side" -anchor "anchor"
13  #Crear y visualizar la etiqueta
14  label [= "LabelName"] -text [= "Text"] -justify "justify"
15  #Crear y visualizar una entry
16  entry [= "EntryName"] -bg "background" -relief "sunken" \
17    -textvariable "var"  -width "width" -justify "justify"
18  pack [= "EntryName"] -side "side" -padx "padx" -pady "pady"
19 }
```

Pseudocódigo 3.9: Procedimiento *Window_bc*.

- Línea 3. Este comando crea una variable, representada por el string *var*, que hará referencia a una imagen del directorio *directory* con nombre *Image*. Esta imagen será la que contenga un botón determinado.
- Línea 5. Esta sentencia crea la ventana para que el usuario asigne las condiciones de contorno. El string *Window* representa el nombre de la ventana en codificación y *Text* es el texto que se mostrará.
- Líneas 7-8. Estos comandos permiten crear y visualizar los marcos que componen la ventana. El string *FrameName* indica el nombre que recibirán cada uno de los marcos y *Text* es el texto que se mostrará. A la hora de visualizarlos *side* indica qué posición ocuparán dentro de la ventana.
- Líneas 10-12. Estos comandos crean y permiten visualizar los botones de los que está compuesta la ventana. El string *ButtonName* indica el nombre con el que se identificarán cada uno de estos botones e *Image* indica la imagen que contendrán. Además, *height* y *width* son las dimensiones que trendrán los botones y *Command* es el comando que se ejecuta cuando se pulsan. Para visualizarlos, *side* y *anchor* son opciones de posición dentro de la ventana.
- Línea 14. Este comando crea una etiqueta que estará contenida en la ventana. El string *LabelName* es el nombre con el que se identificará la etiqueta y *Text* el texto que se mostrará. En este caso *justify* indica el formato que sigue el texto.
- Líneas 16-18. El comando permite crear un campo vacío en el que el usuario puede introducir un valor para la condición de contorno diferente a las especificadas con los botones. El string *EntryName* indica el nombre que tendrá el objeto 'entry', *background* es el color del fondo, *var* representa el nombre de la variable en la que se almacenará la información

que haya en la 'entry' y el resto de strings tienen que ver con la posición dentro de la ventana.

El usuario tiene la opción de elegir uno de los valores numéricos que vienen por defecto o introducir el valor que él quiera. A continuación hay que seleccionar la superficie sobre la que se quiere aplicar. Hay que tener en cuenta que, dependiendo del tipo de entidad geométrica a la que se le estén asignando las condiciones de contorno habrá unas limitaciones, éstas son:

- **Esfera**: en *GiD*® la esfera posee cuatro superficies, mientras que en *NetGen* se representa mediante una sola, por lo tanto, a la hora de asignar las condiciones de contorno, se elija la superficie que se elija en *GiD*®, se asignará a las cuatro el mismo valor.
- **Cilindro**: en el cilindro, la cara lateral viene representada por dos superficies diferentes en *GiD*®, mientras que en *NetGen* es una sola, por ello, la condición de contorno que se asigne a una de las superficies de esa cara lateral, será la que la represente por completo.
- **Cono**: en el caso del cono, a parte de tener la particularidad de la cara lateral, tenemos una representación diferente del mismo en *GiD*® y *NetGen*. En este último lo que se presenta es un tronco de cono, por lo tanto, al no tener disponible esta estructura en *GiD*® no podremos asignale una condición de contorno a su base superior.

El pseudocódigo 3.10 muestra el procedimiento encargado de realizar la asignación de las condiciones de contorno a las superficies de las entidades geométricas simples. Este procedimiento se denomina *Assign_bc* y necesita un parámetro que indica la condición de contorno que hay que asignar a la superficie seleccionada.

```
1 proc Assign_bc {bc} {
2
3   #Identificar la superficie seleccionada
4   set "ID_surface" [GidUtils::PickEntities surfaces single]
5   #Asignar la condicion de contorno
6   GiD_AssignData condition "Condition" Surfaces \
7   "BC" "ID_surface"
8   #Volver a almacenar los datos de los volumenes
9   Button_scape
10 }
```

Pseudocódigo 3.10: Procedimiento *Assign_bc*.

- Línea 4. Este comando permite obtener el identificador de la superficie que el usuario ha seleccionado para asignar una condición de contorno. El string *ID_surface* representa el identificador de la superficie seleccionada.
- Línea 6. Este comando sirve para asignar a la supereficie, con identificador representado por el string *ID_surface*, la condición de contorno dada por el string *BC*. *Boundary* indica el nombre que le hemos asignado a este tipo de condiciones en el fichero *.cnd, cuya explicación se da en la sección 4.3.
- Línea 9. El procedimiento *Button_scape* permite saber si se ha producido algún cambio en el proyecto y cuál ha sido éste para poder actualizar todos los datos que están almacenados en las diferentes listas.

3.6. Paso 5: Gestión de las operaciones.

Las tres operaciones booleanas que están disponibles en el módulo *GiDtoNet* son unión (*or*), intersección (*and*) y sustracción (*and not*). En cada una de ellas están involucradas solamente dos entidades geométricas que el usuario creará y elegirá para formar un nuevo volumen más complejo.

Para abrir la ventana que nos permitirá seleccionar la operación que queremos llevar a cabo accedemos al menú **GiD2Net -> Create volumen**. Esta ventana es la misma que se utilizó para crear las entidades geométricas, veáse figura 3.5. Ahora hay que pulsar el botón de la operación que se quiera realizar y a continuación seleccionar los dos volúmenes implicados en la misma.

El módulo será el encargado de almacenar los identificadores de los dos volúmenes seleccionados para realizar la operación y el tipo de ésta. Con toda la información se ejecutarán una serie de procedimientos que generarán una nueva entidad geométrica a la que llamaremos compuesta.

El procedimiento involucrado en este paso se denomina *Operation* y es el encargado de almacenar la información de las operaciones que se realizan a lo largo del proyecto. Este procedimiento se puede ver en el pseudocódigo 3.11 y a continuación se describen brevemente las líneas que lo componen para que lector comprenda cómo se tratan las operaciones en el módulo *GiDtoNet*.

```
1 proc Operation {op} {
2
3   #Actualizar informacion
4   Button_scape
5
6   #Identificar los volumenes seleccionados
7   set "ID" [GidUtils::PickEntities volumes single]
8
9   #Comprobacion de que los volumenes no son el mismo
10
11  #Almacenar el tipo de operacion
12  set ::GID2NET::LIST_OPERATIONS(TYPE,"Indice") "Operation"
13
14  #Extraer la informacion de los volumenes seleccionados
15  #para realizar la operacion
16
17  #Realizar la operacion y almacenar el volumen obtenido tras ella
18  set ::GID2NET::LIST_OPERATIONS(INFO,"Indice") "ID_volumen1"
19  "Operation" "ID_volumen2"
20  set ::GID2NET::LIST_OPERATIONS(NUM_VOL,"Indice") "LastID"
21  GiD_Process Mescape Geometry Create IntSolid3D "Operation"
22  "ID_volumen1" "ID_volumen2" escape
23  GiD_AssignData condition "Condition" Volumes "Value" "LastID"
24
25  #actualizar informacion
26  Button_scape
27 }
```

Pseudocódigo 3.11: Procedimiento *Operation*.

- Línea 4. Mediante el procedimiento *Button_scape* podemos comprobar si se ha producido algún cambio en el proyecto antes de empezar con la operación y almacenar dichos cambios.
- Línea 7. Este comando permite que podamos obtener en el string *ID* el identificador del volumen que el usuario ha seleccionado para realizar la operación.
- Línea 12. Esta sentencia nos permite almacenar en una lista el tipo de operación que se ha seleccionado. El string *Indice* recoge la posición en la que se va a almacenar esta operación y *Operation* es el tipo de operación.
- Líneas 18-20. Estas sentencias almacenan en una lista toda la información de la operación. El string *Indice* es la posición que ocupa la información de la operación dentro de la lista, *ID_volumen1* e *ID_volumen2* son los identificadores de los volúmenes con los que se realiza la operación y *Operation* es el string con la operación a realizar. Además, *LastID* indica el indentificador que recibirá el volumen obtenido tras la operación.
- Líneas 21-22. Este comando permite que se realice la operación recogida en el string *Operation* entre los volúmenes que poseen los identificadores dados por *ID_volumen1* e *ID_volumen2*.
- Línea 23. Este comando le asigna una condición *GiD*® del tipo *Condition* con valor *Value* al volumen con identificador *LastID*.
- Línea 26. De nuevo, utilizamos el procedimiento *Button_scape* para almacenar los cambios que se han realizado a lo largo del procedimiento.

3.7. Paso 6: Creación del fichero *CSG*.

En esta sección nos detendremos en cómo se genera el fichero *CSG* a partir de las órdenes que da el usuario. Tal y como se ha comentado anteriormente, es en la sección 4.2 en la que se detalla mejor el contenido y la estructura de este fichero.

Para poder generar este fichero se accede al menú **GiD2Net -> Create file *CSG***. El procedimiento encargado de escribir el fichero *CSG* se denomina *Create_file* y se puede encontrar su descripción en el pseudocódigo 3.12. Para generarlo se deben recorrer las listas de almacenamiento de información de las entidades geométricas e ir poco a poco escribiendo las primitivas de los volúmenes que sólo *NetGen* entenderá. Este fichero contiene entidades geométricas simples y compuestas, llevando cada una de ellas unos identificadores que las harán únicas.

```
1 proc Create_file {} {
2
3   #Numero de volumenes actual
4   set "Num_vol" [GiD_Info Geometry NumVolumes]
5   #Ruta del proyecto
6   set "Project" [GiD_Info Project ModelName]
7   #Identificador del proyecto
8   set "ID_file" [open [= "FileName"] w+ 0775]
9   #Primera sentencia del fichero
10  puts "ID_file" "algebraic3d"
11  #Bucle que recorre la informacion de todas las entidades
12  for {set i 0} {$i < "identificadores"} {incr i} {
13    for {set j 0} {$j < "operations"} {incr j} {
14      puts "ID_file" [= "Text"]
15    }
16    puts "ID_file" [= "Text"]
17  }
18
19  #Cerrar fichero
20  close "ID_file"
21 }
```

Pseudocódigo 3.12: Procedimiento *Create_file*.

- Línea 4. Este comando nos permite obtener el número de volúmenes que hay en el proyecto. El string *Num_vol* almacena este valor.
- Línea 6. El string *Project* almacena la ruta en la que se localiza el proyecto que se puede obtener gracias al comando que está a continuación.
- Línea 8. Esta sentencia permite abrir el fichero con nombre *FileName* en modo lectura. El string *ID_file* indica el identificador que se ha dado a este fichero.
- Línea 10. La primera línea para este tipo de ficheros debe contener el string älgebraic3d.
- Líneas 15-20. Se recorren las listas que contienen la información de cada una de las entidades geométricas y se escriben en el fichero según el formato definido por las primitivas de *NetGen*.
- Línea 23. Este comando cierra el fichero con identificador *ID_file*.

3.8. Paso 7: Mallado de las entidades geométricas.

Por último, para generar el mallado de la estructura bajo estudio, el usuario debe acceder al menú **GID2NET -> Generate NetGen mesh**. Éste generará el nuevo mallado mediante una llamada remota a *NetGen*. Las conectividades y la información de la malla obtenida estarán almacenados en un fichero con formato llamado "Neutral Format"(definido por *NetGen* y que se explicará en el capítulo 4) que se conocerá a partir de ahora como fichero *GiD2Net.msh*.

Una vez obtenido este fichero necesitamos hacer un cambio en la estructura y la orientación de los elementos del mismo para que sea posible importar el mallado en *GiD*®. Este cambio es necesario debido a que la orientación que sigue *NetGen* para los tetraedros creados es distinta a la de *GiD*® generando tetraedros con orientación negativa. Un tetraedro tiene orientación negativa cuando el producto vectorial entre el vector 1 (formado por los nodos 1 y 2 del tetraedro) y el vector 2 (formado por los nodos 1 y 3) da como resultado el vector 3 (formado por los nodos 1 y 4) pero con sentido contrario (se pueden ver estos vectores en la figura 3.7). El fichero que se obtendrá tras este cambio se conocerá a partir de ahora como *Net2GiD.msh*.

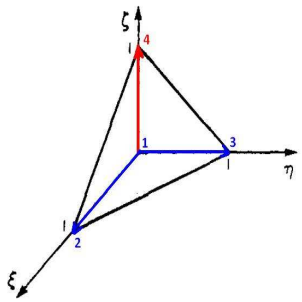


Figura 3.7: Tetraedro de referencia situado sobre sus ejes.

Todo este proceso es totalmente transparente para el usuario, obteniendo finalmente los ficheros *GiD2Net.msh* y *Net2GiD.msh*, siendo este último el que se importe en *GiD*®.

A continuación se presenta el pseudocódigo (ver pseudocódigo) del procedimiento que hace posible la obtención de los dos ficheros que contienen información de mallado, este procedimiento se denomina *Create_meshFile*. Se encarga de realizar la llamada remota a *NetGen*, obteniendo el fichero *GiD2Net.msh*, y de llamar otro procedimiento que se encarga de obtener finalmente el fichero *Net2GiD.msh*. Por último, importa el fichero con el contenido del mallado a *GiD*®.

```
1 proc Create_meshFile {} {
2
3   Escribir_script
4
5   #Ejecutar el fichero para la llamada remota a NetGen
6   exec [file join "directory"] [= "Script"]]
7
8   Create_GiD$^{\begin{small}\textregistered\end{small}}$mesh
9
10  #Obtener el path en el que se encuentra el fichero Net2Gid.msh
11  set "Project_path" [GiD_Info Project ModelName]
12  set [= "FileName"] [file join "Project_path".gid "Net2Gid.msh"]
13
14  #Importar el fichero
15  GiD_Process Mescape Files MeshRead [= "FileName"]
16  GiD_Process Mescape Meshing MeshView escape
17 }
```

Pseudocódigo 3.13: Procedimiento *Create_meshFile*.

- Línea 3. Mediante el procedimiento *Escribir_script* generamos un fichero con extensión .bat que nos servirá para realizar la llamada remota a *NetGen* y generar la malla.
- Línea 6. Este comando permite ejecutar el script creado en la línea anterior y obtener el fichero *.msh con *NetGen*. El string *directory* representa la ruta en la que se encuentra el fichero ejecutable y *Script* indica el nombre de éste.
- Línea 8. Este procedimiento es el encargado de crear el nuevo fichero *.msh a partir del que devuelve *NetGen* con una estructura correcta para ser importado en *GiD*®.
- Líneas 11-12. Permiten almacenar en el string *FileName* la ruta en la que se encuentra el fichero *Net2GiD.msh*.
- Líneas 15-16. Comandos para cargar la nueva malla en *GiD*®. El string *FileName* indica el nombre del fichero que se va a leer.

Para poder llevar a cabo la llamada remota al generador de mallado *NetGen*, es necesario que el módulo cree un fichero ejecutable con extensión *.bat, este fichero se describirá en la sección 4.4. Gracias a este ejecutable podremos hacer la llamada al generador de mallado *NetGen*. El procedimiento que lleva a cabo la creación de este fichero es *Escribir_script* y podemos ver su código en el pseudocódigo 3.14.

```
1 proc Escribir_script {} {
2
3   #Tamanno de malla por defecto en GiD
4   set "mesh_size" [GiD_Info Project RecommendedMeshSize]
5   #Ruta en la que se ubicara el fichero
6   set [= "FileName"] [file join "directory" [= "File.bat"]]
7   #Abrir el fichero
8   set "ID_file" [open   [= "FileName"] w+ 0775]
9   #Ruta en la que se encuentra el ejecutable de NetGen
10  set "Netgen_path" [file join "path" "Netgen-4.9.13_Win32" "bin" "netgen.exe"]
11  #Ruta del proyecto
12  set "Project_path" [GiD_Info Project ModelName]
13  #Ruta en la que se creara el fichero
14  set [= "ProjectName"] [file join "Project_path".gid "GiD2Net.msh"]
15  #Guardar las opciones
16  set options [...]
17  #Escribir en el fichero
18  puts "ID_file" [= "Text"]
19  #Cerrar fichero
20  close "ID_file"
21 }
```

Pseudocódigo 3.14: Procedimiento *Escribir_script*.

- Línea 4. Este comando permite almacenar en el string *mesh_size*, el tamaño de malla que utilizaremos en *NetGen*. Este parámetro es el valor por defecto que tendría una malla en *GiD*®.
- Línea 6. Esta sentencia permite identificar, mediante *FileName*, la ruta en la que queremos almacenar el fichero representado por el string *File.bat*.

- Línea 8. Este comando almacena en *ID_file* el identificador que recibirá el fichero para hacer operaciones con él.
- Línea 12. Esta sentencia almacena en el string *Project_path* la ruta del proyecto actual.
- Línea 16. Se almacenan las opciones que escribiremos en el fichero.
- Línea 18. Mediante este comando escribimos el contenido del fichero. El string *Text* representa lo que vamos a escribir.
- Línea 20. Este último comando permite cerrar el flujo de datos del fichero.

Como ya se ha comentado anteriormente, existe un procedimiento que construirá correctamente el fichero final con la información de mallado de la estructura del proyecto. Este procedimiento se denomina *Create_GiDmesh* y se puede encontrar su contenido en el pseudocódigo 3.15.

```
1 proc Create_GiDmesh {} {
2
3   #Ruta del proyecto
4   set [= "Project_path"] [GiD_Info Project ModelName]
5
6   #Ruta del fichero
7   set [= "File_path"] [file join [= "Project_path"].gid "GiD2Net.msh"]
8
9   #Abrir el fichero en modo lectura
10  set [= "ID_file"] [open [= "File_path"] r 0775]
11
12  #Ruta del fichero
13  set [= "File_path"] [file join [= "Project_path"].gid "Net2Gid.msh"]
14
15  #Abrir el fichero en modo escritura
16  set [= "ID_file"] [open [= "File_path"]  w+ 0775]
17
18  #Escribir en el fichero
19  puts [= "ID_file"] [= "Text"]
20
21  #Cerrar ficheros
22  close [= "ID_file"]
23 }
```

Pseudocódigo 3.15: Procedimiento *Create_GIDMesh*.

- Línea 4. Esta sentencia almacena en el string *Project_path* la ruta en la que se encuentra el proyecto.
- Línea 7. En el string *File_path* se guarda la ruta en la que se encuentra el fichero *GiD2Net.msh* que ha creado *NetGen*.
- Línea 10. Este comando abre el fichero, con identificador *ID_file*, en modo lectura.
- Línea 13. En el string *File_path* se encuentra la ruta en la que vamos a crear el fichero *Net2Gid.msh*.
- Línea 16. Este comando abre el fichero, con identificador *ID_file*, en modo escritura.

- Línea 19. Este comando permite escribir en el fichero con identificador *ID_file* el texto recogido en el string *Text*.
- Línea 22. Comando para cerrar el fichero con identificador *ID_file*.

Una vez desarrollado el proyecto con el módulo *GiDtoNet*, el usuario dispone de una malla generada por *NetGen* pero que está presente en la interfaz de *GiD*®. Una de las ventajas de disponer de esta malla en *GiD*® es que podemos utilizar su herramienta *Mesh Quality* para analizar las características de la malla.

Este estudio será el que nos lleve en el capítulo 5 a comparar el comportamiento de las dos mallas sobre diferentes estructuras, llegando así a una conclusión sobre cuál de las dos mallas proporciona mejores propiedades.

3.9. Conclusiones

En este capítulo hemos realizado una breve explicación de la codificación de los procedimientos más relevantes que intervienen en el módulo *GiDtoNet*.

Hay que tener en cuenta que la aparición de los procedimientos a lo largo del capítulo se ha producido en el orden en el que el usuario los utilizará para realizar su proyecto. Mediante esta estructura y planteamiento de la explicación se ha buscado facilitarle al lector la correcta comprensión del funcionamiento del módulo *GiDtoNet*.

Se recomienda, no obstante, la lectura del apéndice C en el que se encontrará el código completo del módulo para aquellas personas que quieran profundizar más en el análisis del mismo.

Se incluye también en el apéndice A un cuadro resumen con todos los procedimientos del módulo *GiDtoNet* para facilitarle al usuario posibles consultas sobre cuál de ellos realizaba una función concreta.

Capítulo 4

Ficheros involucrados en el uso de *GiDtoNet*

En este capítulo se presentan cada uno de los ficheros que intervienen de alguna manera en el desarrollo de un proyecto con el módulo *GiDtoNet*. Es importante que el usuario conozca su contenido y estructura para comprender mejor el funcionamiento del módulo y para completar la lectura de los capítulos anteriores.

4.1. Introducción.

Para poder explicar de una forma gráfica el contenido de los ficheros que se utilizan en la creación de un proyecto con el módulo *GiDtoNet*, partiremos de un ejemplo para ver el contenido de cada uno de ellos.

En la figura 4.1, se observan una serie de entidades geométricas. Algunos de estos volúmenes se han formado tras realizar una operación, como por ejemplo la unión de las dos esferas de la derecha de la figura. Este proyecto se compone, por lo tanto, de cuatro entidades geométricas simples y dos compuestas.

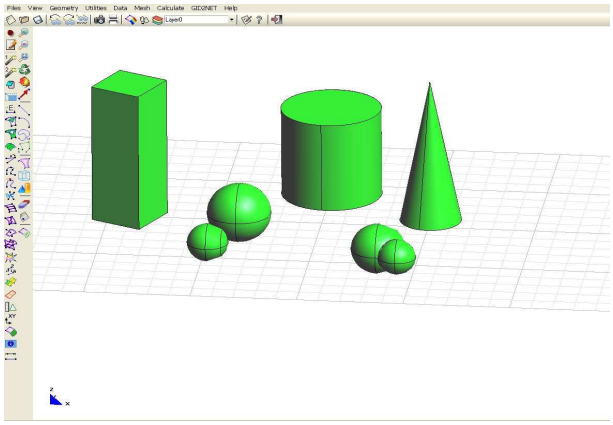


Figura 4.1: Proyecto representado con *GiD*.

4.2. Fichero *CSG*.

El fichero *CSG*, que tendrá extensión *CSG*, contiene las primitivas con la estructura que define *NetGen* para cada uno de los volúmenes que el módulo *GiDtoNet* permite crear. Este fichero debe contener siempre la misma estructura. Cada fichero *CSG* debe comenzar siempre con la palabra clave *algebraic3d* antes de que cualquier línea. La palabra clave *solid*, que acompaña al nombre de la entidad geométrica, define un sólido y a su vez un sólido puede venir definido a partir de las operaciones Eulerianas que sean necesarias, como por ejemplo un hexaedro, como se verá a continuación. Por último, la palabra clave *tlo*, necesaria para el mallado, declara el sólido como *top-level-object*, lo que significa que es un objeto de alto nivel y por lo tanto el que se presentará en la interfaz.

Las primitivas que aporta *NetGen* para poder crear cada uno de los volúmenes que intervienen en un proyecto de un usuario del módulo *GiDtoNet* son:

- **Plano**. El plano es la base para crear tres de los volúmenes soportados por *NetGen*. La estructura de la primitiva que facilita *NetGen* es la siguiente:

  $solid\ plane\_1 = plane(p_x, p_y, p_z; n_x, n_y, n_z);$

  , donde $p = (p_x, p_y, p_z)$ representa un punto cualquiera del plano y $n = (n_x, n_y, n_z)$ representa su normal. Con estos dos elementos, el plano queda totalmente definido.

- **Hexaedro.** Este volumen se define en *NetGen* a partir de la intersección de los seis planos que lo forman. Por lo tanto, la primitiva que identifica en el fichero *CSG* a un hexaedro tendrá la siguiente estructura:

  $solid\ plane\_1 = plane(p_x, p_y, p_z; n_x, n_y, n_z);$
  $solid\ plane\_2 = plane(p_x, p_y, p_z; n_x, n_y, n_z);$
  $solid\ plane\_3 = plane(p_x, p_y, p_z; n_x, n_y, n_z);$
  $solid\ plane\_4 = plane(p_x, p_y, p_z; n_x, n_y, n_z);$
  $solid\ plane\_5 = plane(p_x, p_y, p_z; n_x, n_y, n_z);$
  $solid\ plane\_6 = plane(p_x, p_y, p_z; n_x, n_y, n_z);$
  solid cube = plane_1 and plane_2 and plane_3 and plane_4 and plane_5 and plane_6;
  tlo cube;

- **Esfera.** Para la creación de este volumen, *NetGen* presenta la siguiente primitiva:

  $solid\ sphere\_1 = sphere(c_x, c_y, c_z; r);$
  tlo sphere_1;

  , donde $c = (c_x, c_y, c_z)$ representa el centro de la esfera y r es el radio.

- **Cilindro.** En este caso, el volumen se genera a partir de tres primitivas, dos planos que serán las bases del cilindro y la cara lateral de éste. La primitiva que define *NetGen* para crear un cilindro (lo que en su manual se denomina *cylinder*) solamente crea la cara lateral del mismo, pero en el módulo *GiDtoNet* un cilindro es un sólido que tiene dos bases cerrando la estructura. Por lo tanto la primitiva para este tipo de volúmenes viene definida como:

  $solid\ plane\_1 = plane(p_x, p_y, p_z; n_x, n_y, n_z);$
  $solid\ plane\_2 = plane(p_x, p_y, p_z; n_x, n_y, n_z);$
  $solid\ cyl = cylinder(a_x, a_y, a_z; b_x, b_y, b_z; r);$
  $solid\ cylinder\_1 = cyl\ and\ plane\_1\ and\ plane\_2;$
  $tlo\ cylinder\_1;$

  , donde $a = (a_x, a_y, a_z)$ *y* $b = (b_x, b_y, b_z)$ representan dos puntos del eje central del cilindro mientras que r es el radio. Además tenemos los planos *plane_1* y *plane_2* que son los que forman las bases del cilindro y que se definen mediante las primitivas ya comentadas.

- **Cono.** El caso del cono es un poco más complejo, ya que *NetGen* necesita representar troncos de cono, por lo tanto éste viene definido por la cara lateral y dos planos que actúan de bases para definir el tronco. La estructura que sigue la primitiva para este tipo de volumen es la siguiente:

  $solid\ plane\_1 = plane(p_x, p_y, p_z; n_x, n_y, n_z);$
  $solid\ plane\_2 = plane(p_x, p_y, p_z; n_x, n_y, n_z);$
  $solid\ c = cone(a_x, a_y, a_z; r_a; b_x, b_y, b_z; r_b);$
  $solid\ cone\_1 = c\ and\ plane\_1\ and\ plane\_2;$
  $tlo\ cone\_2;$

  ,donde $a = (a_x, a_y, a_z)$ y $b = (b_x, b_y, b_z)$ son dos puntos del eje central del tronco de cono y r_a y r_b son los radios mayor y menor de éste. Los dos planos que se identifican en la primitiva como *plane_1* y *plane_2* tienen la definición vista anteriormente y actúan como bases mayor y menos del tronco de cono.

Una vez que el usuario tiene en su proyecto las entidades geométricas que quiere mallar puede seleccionar la opción *Create file* CSG para crear este fichero. El módulo *GiDtoNet* genera el fichero *GiD2Net.geo* que será el que utilizará *NetGen* para generar el mallado. Siguiendo con el ejemplo de la figura 4.1 presentamos a continuación el fichero *GiD2Net.geo* creado.

```
algebraic3d

solid p1_0 = plane (-9,0.0,0.0;-0.0,-0.0,-1.0);
solid p2_0 = plane (-8.46,1.15,3.65;0.41,0.9,-0.0);
solid p3_0 = plane (-10.15,0.53,3.65;-0.9,0.41,-0.0);
solid p4_0 = plane (-9.53,-1.15,3.65;-0.41,-0.9,0.0);
solid p5_0 = plane (-7.84,-0.53,3.65;0.9,-0.41,-0.0);
solid p6_0 = plane (-9,0.0,7.31;-0.0,-0.0,1.0);

solid cube_0 = p1_0 and p2_0 and p3_0 and p4_0 and p5_0 and p6_0;
tlo cube_0;

solid hemisphere_1 = sphere (-4.0,1.0,0.0;1.5);
tlo hemisphere_1;

solid p1_2 = plane (0.0,4,4.68;-0.0,-0.0,1.0);
solid p2_2 = plane (0.0,4,0.0;-0.0,-0.0,-1.0);
solid cyl_2 = cylinder(0.0,4,4.68;0.0,4,0.0;2.34);

solid cylinder_2 = cyl_2 and p1_2 and p2_2;
tlo cylinder_2;

solid p1_3 = plane (5,1,0.0;-0.0,-0.0,-1.0);
solid p2_3 = plane (5,1.0,5.06;0.0,0.0,1.0);
solid c_3 = cone (5,1,0.0;1.43;5.0,1.0,7.60;0);

solid cone_3 = c_3 and p1_3 and p2_3;
tlo cone_3;

solid hemisphere_4 = sphere (3.0,-3.0,0.0;1.27);

solid hemisphere_5 = sphere (4.0,-4.0,0.0;0.89);

solid op_1 = hemisphere_4 or hemisphere_5;
tlo op_1;

solid hemisphere_7 = sphere (-5.0,-3.0,0.0;0.96);

solid hemisphere_8 = sphere (-4.0,-3.0,0.0;0.75);

solid op_2 = hemisphere_7 and not hemisphere_8;
tlo op_2;
```

Pseudocódigo 4.1: Contenido del fichero *GiD2Net.geo*.

Por último, vamos a mostrar en la figura 4.2 el aspecto que tendría en *NetGen* el proyecto *GiD*® de la figura 4.1. El usuario nunca accederá a esta vista, ya que llamamos a *NetGen* de forma remota, pero para que quede más clara la finalidad de este fichero y su efectividad veremos cómo están representadas todas las entidades geométricas.

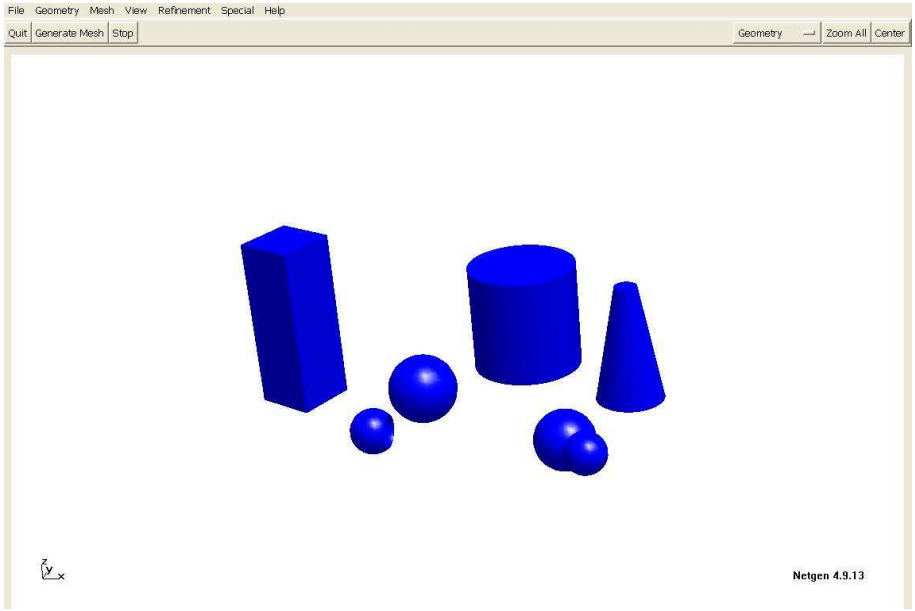


Figura 4.2: Vista de las entidades geométricas del proyecto de la figura 4.1 en *NetGen* utilizando el fichero *GiD2Net.geo*.

4.3. Fichero *.cnd.

Esta sección se va a dedicar a explicar en profundidad el fichero *.cnd. Éste contiene las dos condiciones *GiD*® que se han creado para este módulo. La primera de ellas está destinada a poder identificar el tipo de entidad geométrica y la segunda a asignar las condiciones de contorno. El pseudocódigo 4.2 muestra el contenido de este fichero.

```
1 NUMBER : 1 CONDITION : Type          | NUMBER : 2 CONDITION : Boundary
2 CONDTYPE : over volumes              | CONDTYPE : over surfaces
3 QUESTION : ID                        | QUESTION : ID
4 VALUE : 0                            | VALUE : 0
5 STATE : HIDDEN                       | STATE : HIDDEN
6 END CONDITION                        | END CONDITION
```

Pseudocódigo 4.2: Contenido del fichero *GiDtoNetNet.cnd*.

El código anterior proporciona las características de las dos condiciones *GiD*® que se han creado para trabajar con este módulo. La primera de ellas se describe en la parte izquierda del código y la segunda en la derecha. A continuación detallaremos la primera de las condiciones para conocer un poco mejor las variables de las que está compuesta. La segunda de ellas sólo difiere de la primera en el nombre que se le ha dado y en que se aplica sobre superficies y no sobre volúmenes.

- Línea 1. El string *NUMBER* representa el identificador de la condición *GiD*® y *CONDITION* indica el nombre que le damos a dicha condición, para este primer caso será *Type* porque es el tipo de entidad geométrica mientras que para el segundo será *Boundary*.
- Línea 2. El string *CONDTYPE* indica sobre qué elemento se asignará la condición. La primera de ellas se aplica sobre volúmenes y la segunda sobre superficies, ya que esta última representa las condiciones de contorno de una estructura.
- Línea 3. La palabra *QUESTION* indica qué relacionará *GiD*® con la condición asignada. En ambos casos relacionará los identificadores, volúmenes y superficies respectivamente.

Tipo Entidad Geométrica	Idetificador Condición *GiD*®
Prisma	0
Esfera	1
Cilindro	2
Tronco de cono	3
Operación AND o OR	4
Operación AND NOT	5

Cuadro 4.1: Identificadores para los tipos de entidades geométricas.

4.4. Fichero *.bat.

La llamada remota que se realiza a *NetGen* cuando el usuario le pide al módulo *GiDtoNet* el mallado de la estructura no se podría llevar a cabo sin la ejecución de este fichero. Éste contiene solamente una línea, que es la que permite que se realice esa llamada remota al generador de mallado *NetGen* desde *GiD*®. Cada vez que pedimos el mallado de las entidades geométricas presentes en el proyecto, el módulo se encarga de reescribir el fichero *script.bat*. El pseudocódigo 4.3 contiene el fichero *script.bat*.

```
1 "C:/Archivos de programa/Netgen-4.9.13_Win32/bin/netgen.exe"
2 -geofile = "C:/Documents and Settings/LAURA/Escritorio/E1.gid/Gid2Net.geo"
3 -batchmode -meshfiletype = "Neutral Format" -moderate -meshsize= "Size"
4 -meshfile = "C:Documents and Settings/LAURA/Escritorio/E1.gid/GiDtoNetNet.msh"
```

Pseudocódigo 4.3: Contenido del fichero *script.bat*.

La primera sentencia del archivo es la que permite que *GiD*® haga una llamada al generador de mallado *NetGen*. El resto de opciones indican lo siguiente:

- *-geofile="C:/Documents and Settings/LAURA/Escritorio/E1.gid/Gid2Net.geo"*. Indica la ruta del fichero *GiDtoNetNet.geo* que se utilizará para la llamada a *NetGen*. Esta ruta será distinta en cada proyecto, se ha tomado ésta para ilustrar el ejemplo del contenido del fichero.

- *-batchmode*. Indica que hay que salir del programa una vez generado el mallado.
- *-meshfiletype="Neutral Format"*. Indica el formato del fichero de salida del mallado con extensión *.msh.
- *-meshfile= " C:/Documents and Settings/LAURA/Escritorio/E1.gid/Gid2Net.msh"*. Indica la ruta en la que se creará el fichero *.msh que contendrá la información de mallado.
- *-meshsize*. Controla la medida del mallado. Haremos un mallado lo más fino posible para que las entructuras tengan mayor precisión.

4.5. Ficheros con la información del mallado.

El primero de los ficheros está escrito con el llamado formato neutro (Neutral Format) y su nombre, como ya hemos mencionado, es *GiDtoNet.msh*. Éste se crea tras la llamada remota a *NetGen* y podemos ver su contenido en el pseudocódigo 4.4. Sigue la siguiente estructura:

- **Nodos**. Después del número de nodos que se han creado en el mallado aparece una lista con las coordenadas *x,y,z* de cada uno de ellos.
- **Volúmenes**. Después del número de volúmenes que se han creado tras el mallado, tenemos una lista con todos esos tetraedros. Cada línea del fichero contiene el material al que pertenece el tetraedro y los nodos que lo forman.
- **Superficies**. Después del número de superficies que se han creado al mallar la estructura tenemos un listado con todos los triángulos que forman parte de la superficie exterior de la estructura que se está mallando. Cada línea del fichero en esta sección contiene, la condición de contorno y los identificadores de los tres nodos que forman el triángulo.

```
1 756
2 -10.686750  -0.626506   0.000000
3  -8.373494  -1.686750   0.000000
4  -9.626506   1.686750   0.000000
5
6     ......      ......      ......
7
8   0.125683   2.950746   2.904239
9   0.654178   2.834400   1.222252
10 -0.811970   2.902833   1.287792
11
12 1886
13    1          18         19        131        137
14    1         123         24        142         25
15    1         143        147        132        138
16
17    ....................
18
19    6         161        207        172        180
20    6         752        197        191        194
21    6         163        752        173        174
22
23 1372
24    1           9        123          1
25    1           9          3         24
26    1          16        123          2
27
28    ................
29
30   17         684        685        693
31   17         687        696        695
32   17         690        691        698
```

Pseudocódigo 4.4: Contenido del fichero *GiDtoNetNet.msh* que crea *NetGen*.

Como ya se ha explicado en la sección 3.7, el fichero *GiD2Net.msh* no tiene las mismas conectividades que el *GiDtoNet.msh*, ya que la orientación de los nodos dentro de un tetraedro no es la misma en los dos generadores de mallado. Para obtener una estructura que pueda interpetar *GiD*® y que no genere orientaciones negativas debe sufrir una serie de transformaciones.

Hay que comprobar cada uno de los tetraedros generado en el mallado contenido en el fichero *GiD2Net.msh* para comprobar cuáles de ellos tienen orientación negativa para realizar el cambio en el orden de aparición de sus nodos. Cuando se ha ordenado correctamente la información contenida en el fichero se obtiene el fichero *Net2GiD.msh*. Este fichero se importará automáticamente en *GiD*® y permitirá visualizar la malla que ha creado *NetGen*. Su contenido se divide en las siguientes partes:

- **Nodos**. Tras el string *Coordinates* encontramos las coordenadas *x,y,z* de cada uno de los nodos del mallado.

- **Volúmenes**. Tras el string *Elements* podemos ver qué nodos forman cada uno de los tetraedros. Las líneas que recogen los nodos que forma un tetraedro comienzan con el número de dicho elemento.

```
1 MESH     dimension 3 ElemType Tetrahedra  Nnode 4
2 Coordinates
3   1 -10.686750  -0.626506 0.000000
4   2 -8.373494 -1.686750 0.000000
5   3 -9.626506 1.686750  0.000000
6
7   ..............
8
9   754 0.125683  2.950746  2.904239
10  755 0.654178  2.834400  1.222252
11  756 -0.811970 2.902833  1.287792
12 end coordinates
13
14 Elements
15  1    19    137   18    131
16  2    24    25    123   142
17  3    147   138   143   132
18
19  ..............
20
21  1884    207   180   161   172
22  1885    197   194   752   191
23  1886    752   174   163   173
24 end elements
```

4.6. Conclusiones

Se ha considerado importante que el lector pueda comprender el contenido y la finalidad de todos los ficheros presentes en un proyecto en el que interviene el módulo *GiDtoNet* para ser capaz de utilizarlo correctamente.

En el apéndice B se puede encontrar una tabla resumen con una breve descripción de cada uno de los ficheros que intervienen en un proyecto desarrollado mediante el módulo *GiDtoNet*.

Ahora que el usuario, tras la lectura del presente Proyecto Fin de Carrera, tiene un conocimiento total de la estructura y de los pasos que hay que seguir en la ejecución del módulo *GiDtoNet*, se le considera listo para interaccionar con él libremente, pudiendo crear cualquier estructura que desee.

Capítulo 5

Batería de pruebas

A lo largo de este capítulo se realizarán una serie de pruebas que servirán para comparar los dos generadores de mallado bajo estudio. Mediante la realización de esta comparativa esperamos descubrir cuál de ellos presenta mejores características bajo las mismas condiciones de configuración.

El generador de mallado *GiD*® nos aporta una serie de parámetros de medida de calidad de las mallas que utilizaremos para hacer nuestras comparaciones. Además de los parámetros que proporciona *GiD*®, vamos a utilizar otras dos medidas que consideramos importantes para evaluar un generador de mallado, tales como el número de elementos que componen la malla y el tiempo que tarda en realizarse.

Obviamente, el espacio muestral del que disponemos puede ser infinito, por lo que sólo llevaremos a cabo aquellos proyectos que resulten más completos y que nos permitan obtener resultados lo suficientemente fiables como para exponer unas conclusiones sólidas.

5.1. Introducción

Lo primero que hay que comprobar para poder comparar *GiD*® y *NetGen* es que los parámetros de malladado de los que disponen cada uno de ellos estén configurados de tal forma que el comportamiento de ambos generadores de mallado actúen igual. Estos parámentros de configuración se detallaron en el capítulo 2.

A continuación se listan todos los parámetros de configuración de los generadores de mallado que se modificarán para realizar las pruebas. Algunos de ellos van a permanecer constantes para que los dos generadores de mallado actúen de la misma forma.

- Tamaño de los elementos: este valor será distinto en función de la prueba para observar el comportamiento con diversos valores.
- Orden de los elementos: los elementos en todos los casos bajo estudio serán de orden dos.

- Grading: este valor será 0.3 o 0.6 en función de la prueba. Hemos tomado esos dos valores porque son unos valores intermedios dentro del rango permitido (entre 0 y 1) para apreciar las diferencias sobre diferentes mallados en función de este parámetro.
- RJump: en el caso de *GiD*® se ha elegido esta forma de mallar ya que es la misma que la de *NetGen*.

La batería de pruebas consistirá en analizar primero las cuatro entidades geométricas simples que se pueden crear con el módulo *GiDtoNet* y a continuación dos estructuras en los que están involucradas varias entidades geométricas, ya sean simples o compuestas.

A partir de la ejecución de estas pruebas obtendremos una serie de conclusiones que quedarán recogidas en el capítulo 6 y que determinarán qué generador de mallado tiene mejores propiedades.

Los parámetros que mediremos para determinar la calidad de los mallados son:

1. **Número de elementos del mallado.** Este parámetro de calidad indica el número de tetraedros que forman la malla que se crea, tanto con *GiD*® como con *NetGen*. Lo importante de esta medida es que una estructura con un mallado compuesto por muchos elementos hace que el tiempo de procesado, mediante métodos numéricos, de la estructura aumente. A su vez, las matrices que se utilizan para estudiar estos elementos con métodos numéricos crecen en dimensiones y se hace complicado su análisis.

2. **Tiempo en la realización del mallado.** Con este páramentro de calidad vamos a saber el tiempo que tardan *GiD*® y *NetGen* en realizar sus respectivos mallados sobre la misma estructura. Se prefiere un generador de mallado que genere los mallados en el menor tiempo posible sin perder precisión. Estas medidas de tiempo se realizarán con un ordenador con procesador Intel® Core2 Duo con 1 GB de RAM y CPU T5670 a 1.80 GHz.

3. **Mínimo tamaño de los ángulos de los tetraedros ("Minimum dihedral angle").** Este criterio de calidad mide el ángulo interior mínimo de los tetraedros de la malla. Este parámetro permite distinguir si un tetraedro está deformado o se parece al de referencia. Se considera que la calidad de un tetraedro disminuye cuando el valor de este ángulo es inferior a 20 grados.

4. **Máximo tamaño de los ángulos de los tetraedros ("Maximum dihedral angle").** Este parámetro mide el ángulo interior máximo de los tetraedros de la malla y el estudio es complementario al que mide el ángulo mínimo. Generalmente se utiliza mucho más el criterio del mínimo ángulo para clasificar triángulos y tetraedros y el del máximo ángulo para cuadriláteros y hexaedros. En este caso, el valor del parámetro en cada uno de los tetraedros se recomienda que sea menor de 120 grados para evitar que esté deformado, en caso contrario se considera que el tetraedro tiene peor calidad.

5. **Forma de los tetraedros ("Shape quality").** Este parámetro de mallado se encarga de medir la relación entre la forma de los tetraedros de la malla y el tetraedro de referencia. Cuanto más se aproximen a su forma, más cerca estará de uno el valor de este parámetro. La expresión matemática para este tipo de medidas es

$$q = \frac{6\sqrt{2}\text{Volume}}{\sum_{i=1}^{3} l_i^3}$$

, donde *'Volume'* se refiere al volumen del tetraedro y *li* indica el tamaño tamaño de los bordes de los tetraedros [2].

6. **Volumen de los elementos ("Element vol").** Este parámetro de calidad del mallado proporciona el volumen de cada uno de los elementos que forman la malla. Se puede relacionar directamente con el número de elementos que la componen de forma que a mayor número de elementos, menor será el volumen de cada uno de ellos. Este valor nos servirá para comprobar la veracidad del número de elementos de la malla, ya que es un valor que la herramienta de mallado no proporciona de forma directa al cargar la malla creada con el módulo *GiDtoNet*.

5.2. Mallado de una esfera.

Esta prueba consiste en realizar el mallado de una esfera de radio uno, representada en las figuras 5.1 y 5.2. Este mallado se llevará a cabo con los dos generadores de mallado bajo estudio para comprobar cuál de estas mallas posee mejores propiedades. La esfera es una estructura un poco especial ya que su superficie es curva, lo que hace que la malla deba adaptarse muy bien a ella.

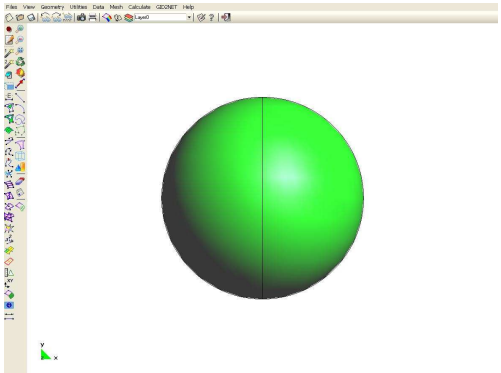


Figura 5.1: Estructura de una esfera con *GiD*.

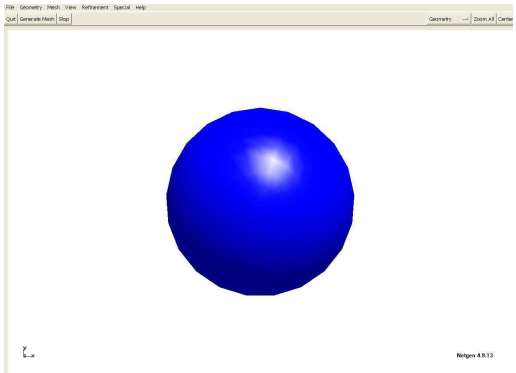


Figura 5.2: Estructura de una esfera con *NetGen*.

Los parámetros que se han utilizado para su realización son tamaño de malla 0.6 y grading 0.3, el resto de las opciones de configuración son las comentadas en la introducción del presente capítulo. En las figuras 5.3 y 5.4 se puede observar el mallado obtenido con *GiD*® y *NetGen*. A simple vista se puede apreciar que la malla de la superficie exterior es muy similar en los dos casos, eso se debe a la configuración que hemos realizado de los dos generadores de mallado.

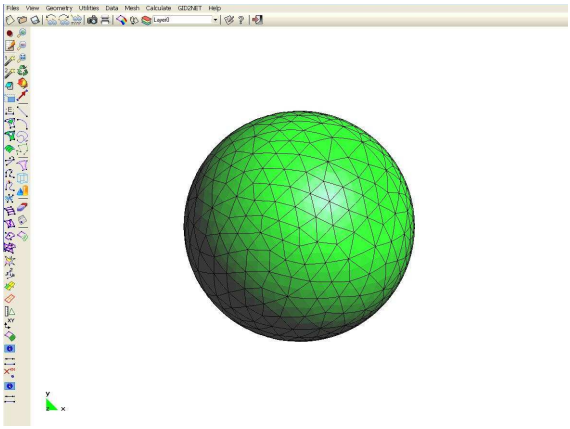


Figura 5.3: Mallado de una esfera con *GiD*.

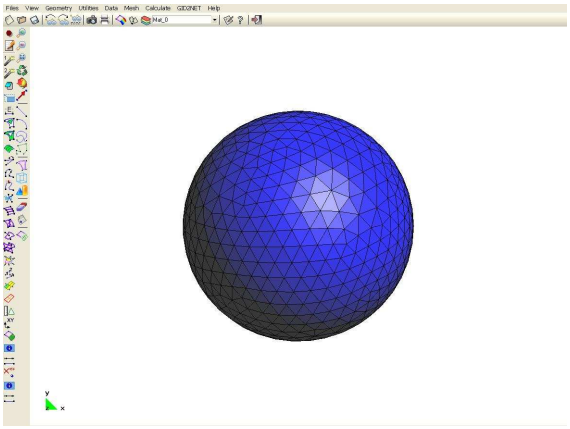


Figura 5.4: Mallado de una esfera con *NetGen*.

5.2.1. Número de elementos del mallado.

A pesar de tener una estructura exterior muy parecida, los mallados generados por *GiD*® y *NetGen* tienen distinto número de elementos.

Si mallamos la esfera con *GiD*® se obtienen 13341 tetraedros mientras que si la malla se realiza con *NetGen* tiene 4488, alrededor de una tercera parte. El hecho de tener más tetraedros supone que al estudiar la estructura mediante métodos numéricos se tengan más incógnitas y aumente el tiempo de procesado de los datos. En este caso los resultados son mejores si mallamos con *NetGen*, por lo tanto la malla que proporciona el módulo *GiDtoNet* es mejor que la proporcionada directamente con *GiD*®.

5.2.2. Tiempo en la realización del mallado.

A continuación se ha medido el tiempo que tardan ambas herramientas en generar el mallado de la estructura bajo estudio.

Para este caso en concreto hemos obtenido que *GiD*® tarda en realizar la malla 3.881 s y *NetGen* 3.703 s. Con lo que tenemos una diferencia de unos 176 ms entre *GiD*® y *NetGen*.

Esta diferencia apenas es perceptible por el usuario porque en ambos casos estamos hablando de cerca de 3.8 s en la realización del mallado. Aún así, en este sentido mejoramos ligeramente los resultados utilizando *NetGen*, ya que tarda alrededor de 0.2 s menos en realizar la malla. Si este valor se escala en función del número de elementos, puede que tenga más sentido elegir entre un generador de mallado u otro, pero con estos valores se deja que sea el usuario el que decida cuál de ellos prefiere.

5.2.3. Mínimo ángulo de los tetraedros.

Este parámetro, como ya se ha comentado, mide el ángulo interior mínimo de los tetraedros que componen la malla. En las gráficas 5.5 y 5.6 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea menor de 20 grados tienen peor calidad, por lo tanto el que más elementos tenga por debajo de este valor proporcionará una malla con unas características peores.

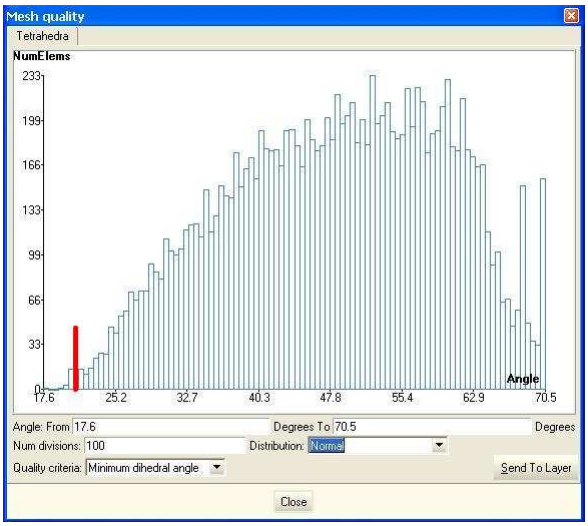


Figura 5.5: Tamaño mínimo del ángulo de los tetraedros con *GiD*.

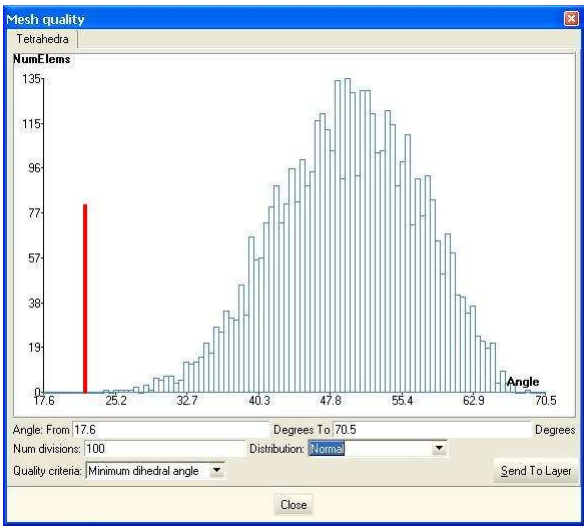


Figura 5.6: Tamaño mínimo del ángulo de los tetraedros con *NetGen*.

Como se puede observar en las gráficas 5.5 y 5.6, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros comienzan a ser peores. Si se cuenta el número de tetraedros que hay por debajo de este umbral en uno y otro caso tenemos los siguientes resultados:

- *GiD*®: 23 tetraedros. Que obteniendo el porcentaje respecto al número total de tetraedros que tiene la malla se obtiene un 0.173 %.
- *NetGen*: 0 tetraedros. En este caso no hay ninguno que esté por debajo del umbral.

Una vez estudiados los valores numéricos recogidos, se comprueba que en este caso, la malla proporcionada por *NetGen* es mejor que la que genera *GiD*®.

5.2.4. Máximo ángulo de los tetraedros.

Este parámetro mide el ángulo interior máximo de los tetraedros de una malla malla. En las gráficas 5.7 y 5.8 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea mayor de 120 grados tienen peor calidad, por lo tanto el que más elementos tenga por encima de este valor proporcionará una malla con unas características peores.

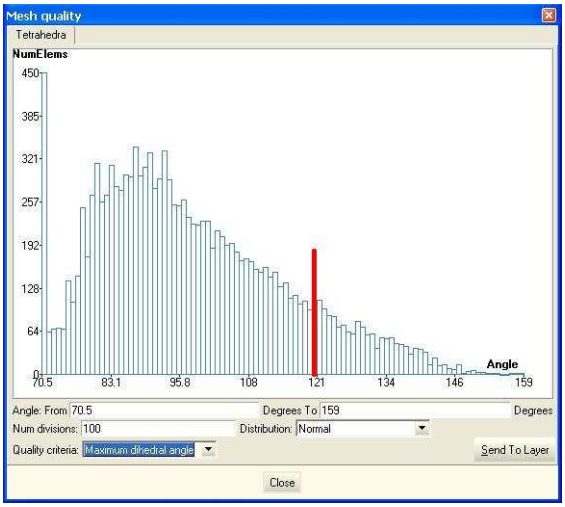


Figura 5.7: Tamaño máximo del ángulo de los tetraedros con *GiD*.

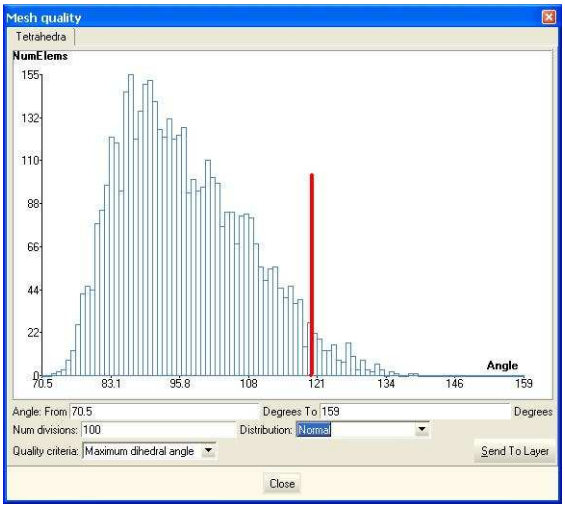


Figura 5.8: Tamaño máximo del ángulo de los tetraedros con *NetGen*.

Como se observa en las gráficas 5.7 y 5.8, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros tienen peor calidad. Si se cuenta el número de tetraedros que hay por encima de este umbral en uno y otro caso se obtienen los siguientes resultados:

- *GiD*®: 2174 tetraedros. Que obteniendo el porcentaje respecto al número total de éstos que tiene la malla se obtiene un 16.29 %.
- *NetGen*: 198 tetraedros. Mediante la misma operación anterior se obtiene un 4.41 % repecto al número total de tetraedros del mallado.

Gracias a los resultados anteriores, se puede afirmar que la malla generada por *NetGen* tiene mejor calidad que la proporcionada por *GiD*®, ya este último tiene un mayor porcentaje de tetraedros de peor calidad en su malla.

5.2.5. Forma de los tetraedros.

En las gráficas 5.9 y 5.10 están representados los resultados obtenidos para el mallado con *GiD*® y *NetGen*. Este parámetro de calidad se encarga de medir la relación que existe entre los tetraedros de la malla y el de referencia. Cuanto más se parezca un tetraedro al de referencia más cercano a uno estará el valor medido en este caso.

Si nos fijamos en los resultados obtenidos, se tiene que para la mallada creada con *GiD*® los valores están comprendidos entre 0.4725 y 1, mientras que para el caso de *NetGen* los valores se encuentran entre 0.66 y 0.9515. La mayoría de los elementos en el caso de *GiD*® tienen una semejanza con el tetraedro de referencia de 1, son prácticamente iguales, mientras que para el caso de *NetGen* ese valor es 0.854.

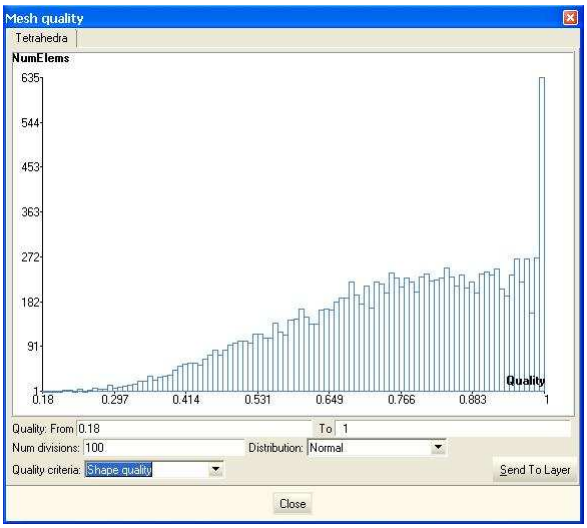


Figura 5.9: Forma de los tetraedros con *GiD*.

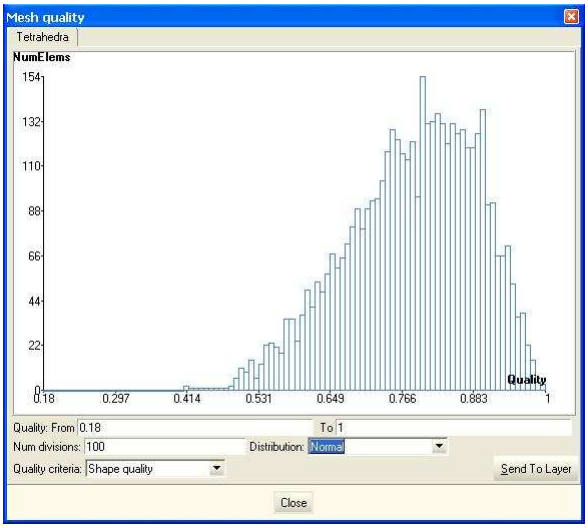


Figura 5.10: Forma de los tetraedros con *NetGen*.

Tras observar los valores obtenidos en ambas gráficas, podemos afirmar que los tetraedros generados con *NetGen* tienen una forma con una relación más estable respecto al de referencia, ya que aunque no exactamente iguales a éste, posee una forma cercana a la del tetraedro de referencia. Sin embargo, aunque *GiD*® tiene muchos de ellos centrados en el valor uno, tiene otros muchos muy alejados de la forma que se toma como referencia.

5.2.6. Volumen de los elementos.

Las gráficas 5.11 y 5.12 recogen el volumen de los elementos que forman las mallas creadas con cada uno de los generadores de mallado bajo estudio. Este parámetro está directamente relacionado con el número de elementos, ya que a mayor número de ellos, menor será su volumen.

Si mallamos con *GiD*®, los valores de la gráfica están comprendidos entre 0.01025 y 0.0327, y en el caso de mallar con *NetGen* entre 0.022723 y 0.122. Además, en el caso de *GiD*®, la mayoría de sus elementos tienen un volumen con un valor en torno a 0.017735, mientras que con *NetGen* se encuentran alrededor de 0.034.

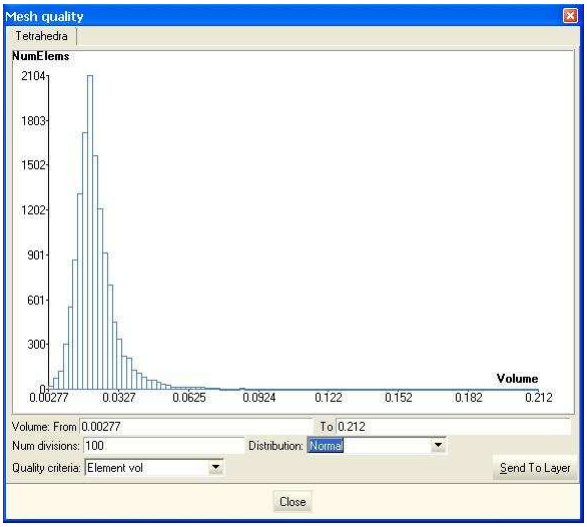


Figura 5.11: Volumen de los elementos con *GiD*.

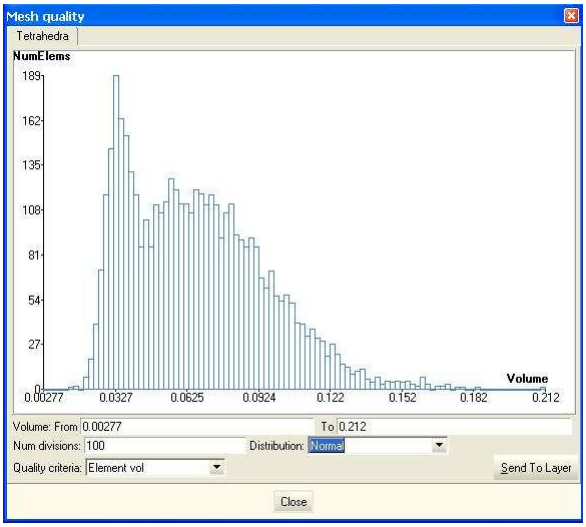


Figura 5.12: Volumen de los elementos con *NetGen*.

Tras analizar los resultados recogidos en las gráficas anteriores, se cumple la relación existente entre el número de elementos de la malla, que se ha medido anteriormente, y el tamaño de los elementos que la forman. Esta relación es inversamente proporcional y por lo tanto, con *GiD*® se obtienen elementos con volúmenes más pequeños, ya que el número de tetraedros que componen su malla es mayor.

5.2.7. Conclusiones.

Tras haber estudiado detenidamente cada una de las gráficas que miden de forma cualitativa el mallado de esta estructura en cuestión con la misma configuración en ambos generadores de mallado, podemos afirmar que los resultados obtenidos con *NetGen* son mejores que con *GiD*®. La malla proporcionada por *NetGen* tiene posee mejores características. Por lo tanto, para un caso similar se recomienda que usuario utilice el mallado proporcionado por el módulo *GiDtoNet* antes que el obtenido directamente con *GiD*®.

A continuación se muestra una tabla en la que se recogen los resultados obtenidos para cada una de las medidas de calidad realizadas en los apartados anteriores. De esta forma, lector tendrá una visión global de los valores que se manejan y de por qué realizamos las afirmaciones sobre cuál de los dos generadores de mallado presenta mejores resultados.

	GiD®	**NetGen**
Número de tetraedros	13341	4488
Tiempo generación mallado (s)	3.881	3.703
Ángulo mínimo < 20 grados	23 tetraedros	0 tetraedros
% respecto al total	0.173 %	0 %
Ángulo máximos > 120 grados	2174 tetraedros	198 tetraedros
% respecto al total	16.29 %	4.41 %
Forma de los tetraedros	1	0.854
Volumen de los elementos	0.017735	0.034

5.3. Mallado de un hexaedro.

La segunda prueba que se ha creado para analizar las propiedades de los mallados facilitados con las dos herramientas bajo estudio consiste en un hexaedro irregular que se encuentra representado en las figuras 5.13 y 5.14 junto con sus dimensiones. Ésta es otra de las entidades geométricas simples que *GiDtoNet* permite construir ya que está soportada por *NetGen*.

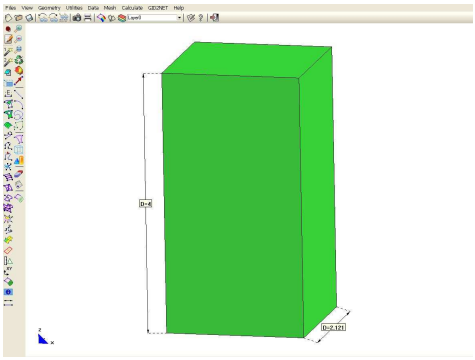


Figura 5.13: Estructura de un hexaedro con *GiD*.

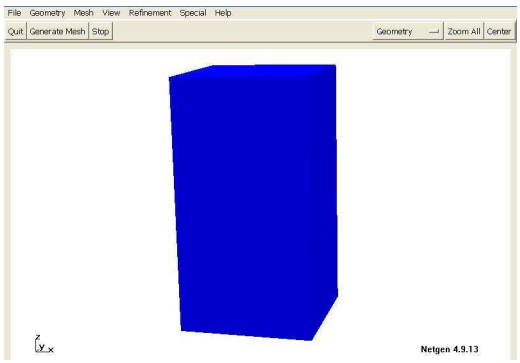


Figura 5.14: Estructura de un hexaedro con *NetGen*.

En las figuras 5.15 y 5.16 se observan los mallados obtenidos con ambas herramientas. Para la creación de estas mallas se ha utilizado un tamaño de mallado de 0.3 y un grading de 0.6 para ver los resultados con otros valores de estos dos parámetros. El resto de opciones de configuración en ambos generadores de mallado permanecen con los mismos valores citados en la introducción del presente capítulo. A simple vista se puede apreciar que la malla de la superficie exterior es muy similar en los dos casos, eso es debido a la configuración que hemos elegido de los dos generadores de mallado.

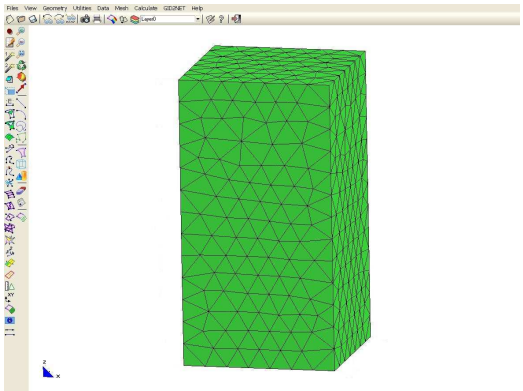


Figura 5.15: Mallado de un hexaedro con *GiD*.

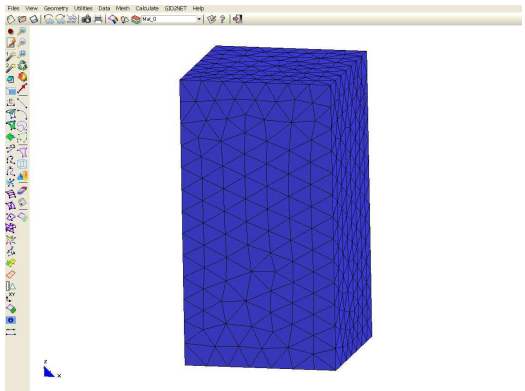


Figura 5.16: Mallado de un hexaedro con *NetGen*.

5.3.1. Número de elementos en el mallado.

A pesar de tener una estructura exterior muy parecida, los mallados generados por *GiD*® y *NetGen* tienen distinto número de elementos.

Al mallar el hexaedro con *GiD*® se obtienen 6295 tetraedros mientras que si la malla se realiza con *NetGen* se obtienen 2845. El hecho de tener más tetraedros supone que al estudiar la estructura mediante métodos numéricos se tengan más incógnitas y aumente el tiempo de procesado de los datos. En este caso los resultados son mejores si mallamos con *NetGen*, por lo tanto la malla que proporciona el módulo *GiDtoNet* es mejor que la proporcionada directamente con *GiD*®.

5.3.2. Tiempo en la realización del mallado.

Como se ha comentado en la introducción del presente capítulo, este parámetro de calidad consiste en medir el tiempo que tardan ambas herramientas en generar el mallado de la estructura bajo estudio.

En este caso en concreto hemos obtenido que *GiD*® tarda en realizar la malla 1.189 s y *NetGen* 2.234 s. Con lo que tenemos una diferencia de unos 1.045 s entre *GiD*® y *NetGen*. A diferencia del caso anterior, estos valores si qué permiten elegir entre un mallador y otro, ya que el usuario puede percibir que *NetGen* tarda más en crear el mallado, alrededor de 1 s. Por lo tanto, basándonos en este parámetro, se le recomendaría al usuario la utilización de *GiD*® para llevar a cabo el mallado de la estructura.

5.3.3. Mínimo ángulo de los tetraedros.

Este parámetro, como ya se ha comentado, mide el ángulo interior mínimo de los tetraedros que componen la malla. En las gráficas 5.17 y 5.18 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea menor de 20 grados tienen peor calidad, por lo tanto el que más elementos tenga por debajo de este valor proporcionará una malla con unas características peores.

Como se puede observar en las gráficas 5.17 y 5.18, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros comienzan a tener peor calidad. Si se cuenta el número de tetraedros que hay por debajo de este umbral en uno y otro caso tenemos los siguientes resultados:

- *GiD*®: 12 tetraedros. Que obteniendo el porcentaje respecto al número total de tetraedros que tiene la malla se obtiene un 0.1906 %.

- *NetGen*: 0 tetraedros. En este caso no hay ninguno que esté por debajo del umbral.

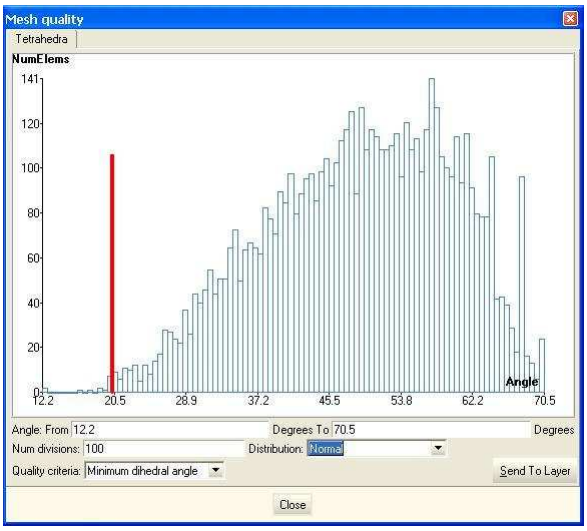


Figura 5.17: Tamaño mínimo del ángulo de los tetraedros con *GiD*.

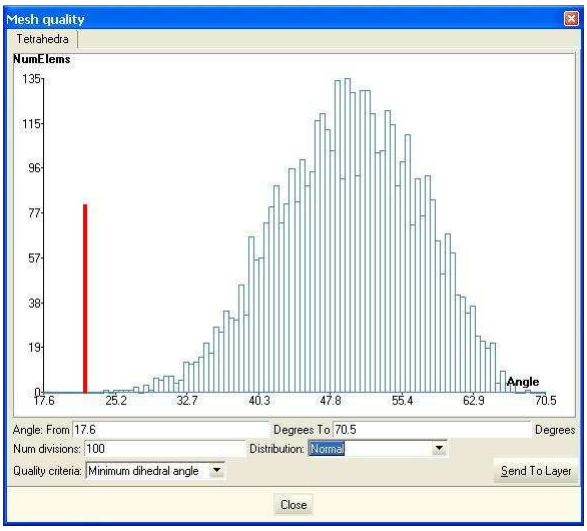


Figura 5.18: Tamaño mínimo del ángulo de los tetraedros con *NetGen*.

Una vez estudiados los valores numéricos recogidos, se comprueba que en este caso, la malla proporcionada por *NetGen* es ligeramente mejor que la generada por *GiD*®.

5.3.4. Máximo ángulo de los tetraedros.

Este parámetro mide el ángulo interior máximo de los tetraedros de una malla malla. En las gráficas 5.19 y 5.20 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea mayor de 120 grados tienen peor calidad, por lo tanto el que más elementos tenga por encima de este valor proporcionará una malla con unas características peores.

Como se observa en las gráficas 5.19 y 5.20, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros dejan de ser buenos. Si se cuenta el número de tetraedros que hay por encima de este umbral en uno y otro caso se obtienen los siguientes resultados:

- *GiD*®: 723 tetraedros. Que obteniendo el porcentaje respecto al número total de éstos que tiene la malla se obtiene un 11.485 %.
- *NetGen*: 20 tetraedros. Mediante la misma operación anterior se obtiene un 0.703 % repecto al número total de tetraedros del mallado.

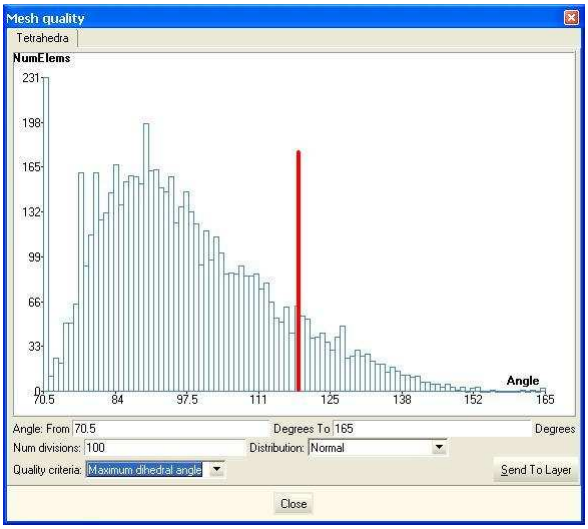


Figura 5.19: Tamaño máximo del ángulo de los tetraedros con *GiD*.

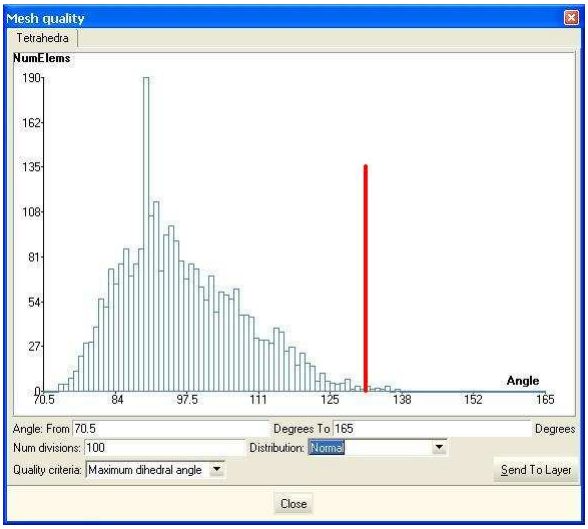


Figura 5.20: Tamaño máximo del ángulo de los tetraedros con *NetGen*.

Gracias a los resultados anteriores, se puede afirmar que la malla que genera *NetGen* tiene mejor calidad que la proporcionada por *GiD*®, ya que este último tiene un mayor porcentaje de tetraedros de peor calidad en su malla.

5.3.5. Forma de los tetraedros.

En las gráficas 5.21 y 5.22 están representados los resultados obtenidos para el mallado con *GiD*® y *NetGen*. Este parámetro de calidad se encarga de medir la relación que existe entre los tetraedros de la malla y el de referencia. Cuanto más se parezca un tetraedro al de referencia más cercano a uno estará el valor medido en este caso.

Si nos fijamos en los resultados obtenidos, se tiene que para la mallada creada con *GiD*® los valores están comprendidos entre 0.607 y 1, mientras que para el caso de *NetGen* los valores se encuentran entre 0.666 y 0.956. La mayoría de los elementos en el caso de *GiD*® tienen una semejanza con el tetraedro de referencia de 1, son prácticamente iguales a éste, mientras que para el caso de *NetGen* ese valor es 0.869.

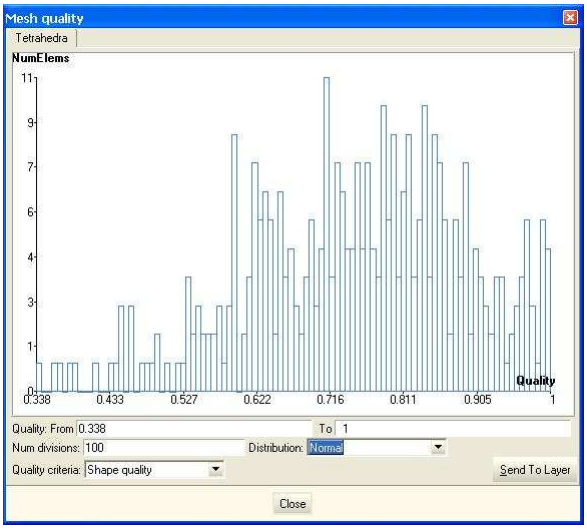


Figura 5.21: Forma de los tetraedros con *GiD*.

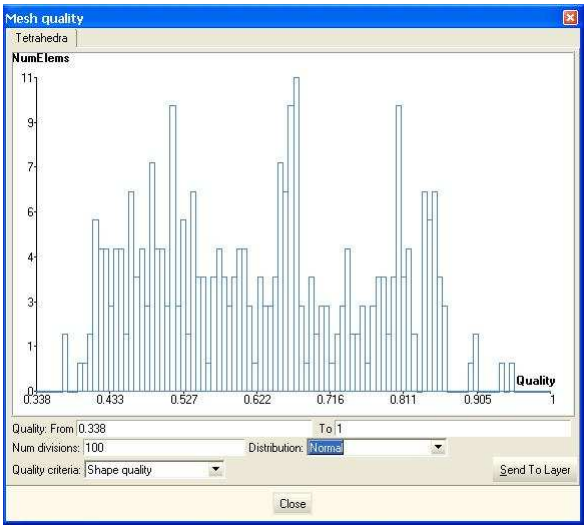


Figura 5.22: Forma de los tetraedros con *NetGen*.

Tras observar los valores obtenidos en ambas gráficas, podemos afirmar que los tetraedros generados con *NetGen* tienen una forma con una relación más estable respecto al de referencia, ya que aunque no tienen una forma exactamente igual a este tetraedro, poseen una forma cercana a referencia. Sin embargo, aunque *GiD*® tiene muchos de ellos centrados en el valor uno, tiene otros muchos muy alejados de la forma de ese tetraedro que tomamos como referencia.

5.3.6. Volumen de los elementos.

Las gráficas 5.23 y 5.24 recogen el tamaño de los tetraedros que componen la malla que se ha creado con ambos generadores de mallado. Este parámetro nos permite corroborar que el valor obtenido para el número de elementos de la malla tiene sentido. Cuantos más elementos tengamos en el mallado menor será el volumen de cada uno de ellos.

Si mallamos con *GiD*®, los valores de la gráfica están comprendidos entre 0.00166 y 0.00414, y en el caso de mallar con *NetGen* entre 0.002665 y 0.01175. Además, en el caso de *GiD*®, la mayoría de sus elementos tienen un volumen con un valor en torno a 0.00249, mientras que con *NetGen* se encuentran alrededor de 0.00493, prácticamente el doble que en el primer caso.

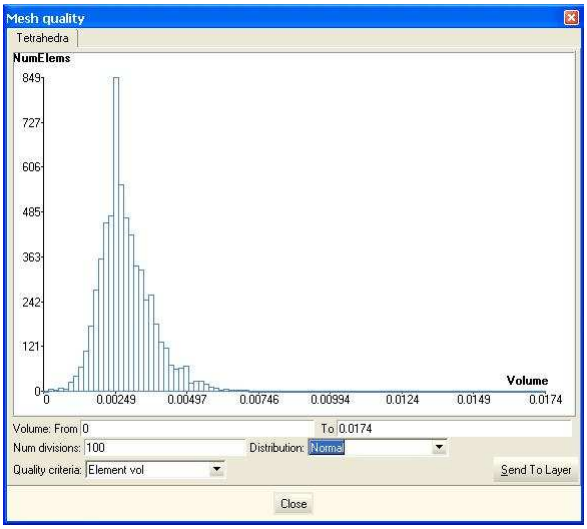


Figura 5.23: Volumen de los elementos con *GiD*.

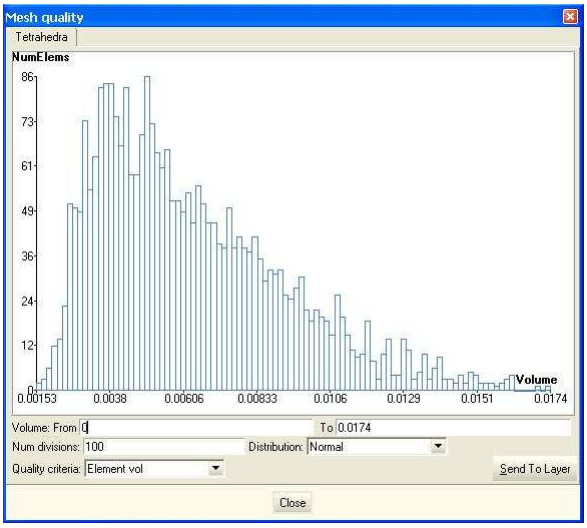


Figura 5.24: Volumen de los elementos con *NetGen*.

Tras estudiar los resultados obtenidos con ambos generadores de mallado para este parámetro de calidad se comprueba que, como era de esperar, se obtienen tetraedros de tamaño mayor con *NetGen* porque se ha generado una malla que tiene menos elementos.

5.3.7. Conclusiones.

Tras haber realizado un estudio de la malla creada con los dos generadores bajo estudio, tenemos que afirmar que se obtienen mejores resultados utilizando el módulo *GiDtoNet* que proporciona el mallado generado con *NetGen*.

A pesar de que *NetGen* tienen un tiempo de realización del mallado de un segundo más que *GiD*®, crea una malla que posee muchos menos elementos. Por este motivo, a la hora de realizar cálculos numéricos con la malla creada, el tiempo de procesado de los resultados con la obtenida con *GiD*® será mucho mayor, siendo más importante que el tamaño del sistema de ecuaciones que hay que resolver sea más pequeño.

A continuación se muestra una tabla en la que se recogen los resultados obtenidos para cada uno de los estudios de calidad realizados. Estos datos nos permiten tener una visión global de las pruebas realizadas en los puntos anteriores sobre cada una de las mallas bajo estudio.

	GiD®	**NetGen**
Número de tetraedros	6295	2845
Tiempo generación mallado (s)	1.189	2.234
Ángulo mínimo < 20 grados	12 tetraedros	0 tetraedros
% respecto al total	0.1906 %	0 %
Ángulo máximos > 120 grados	723 tetraedros	20 tetraedros
% respecto al total	11.485 %	0.703 %
Forma de los tetraedros	1	0.869
Volumen de los elementos	0.00249	0.00493

5.4. Mallado de un cilindro.

En este caso se ha desarrollado una prueba que consiste en crear y mallar un cilindro mediante *GiD®* y *NetGen*. Las dimensiones de éste se recogen en la figura 5.25. Además se ha introducido otra figura (véase figura 5.26) que representa el volumen en la herramienta *NetGen*. Esta estructura también posee una característica importante, su superficie exterior es curva, por lo que los tetraedros que se creen en el mallado deben adaptarse muy bien a ella.

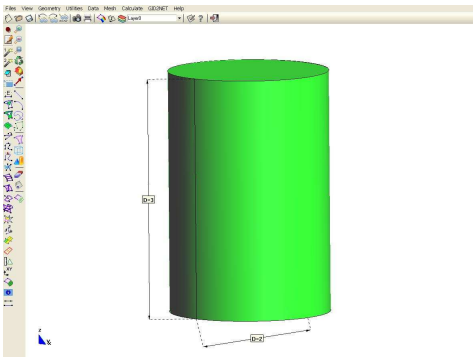


Figura 5.25: Estructura de un cilindro con *GiD*.

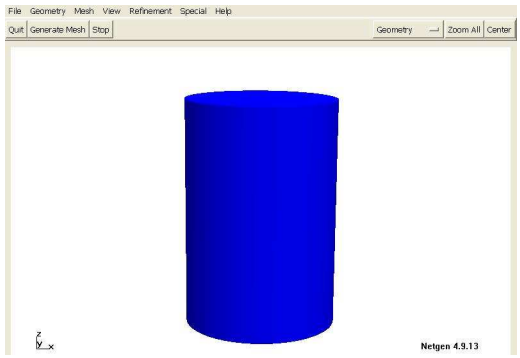


Figura 5.26: Estructura de un cilindro con *NetGen*.

Para realizar el mallado de la estructura en cuestión se ha optado por utilizar un tamaño de malla de 0.6 y un grading de 0.6, siendo el resto de opciones de configuración las mismas que se han utilizado hasta ahora y detalladas en la introducción de este capítulo. En las figuras 5.27 y 5.28 se observan los mallados obtenidos con ambos generadores de mallado.

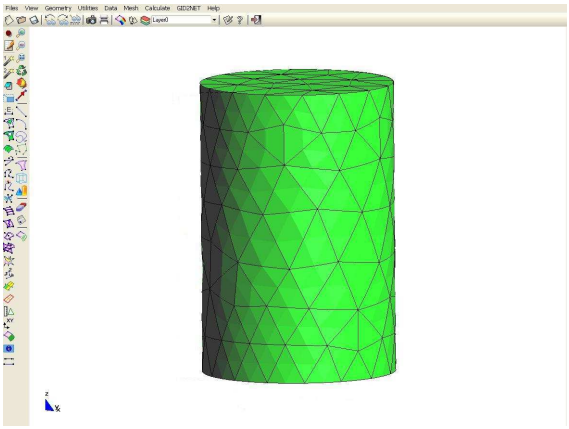


Figura 5.27: Mallado de un cilindro con *GiD*.

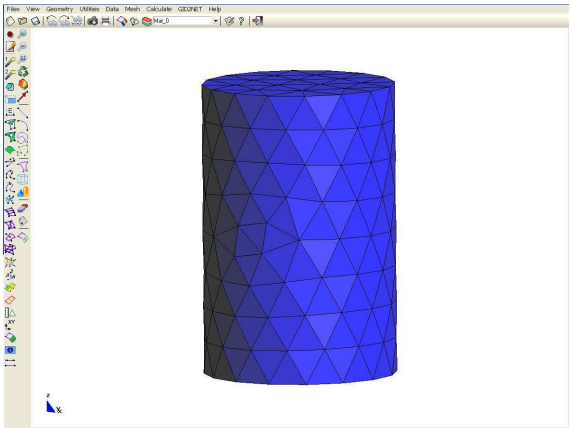


Figura 5.28: Mallado de un cilindro con *NetGen*.

5.4.1. Número de elementos en el mallado.

A simple vista se puede apreciar que el mallado superficial en ambos es ligramente distinto. De hecho, parece que hay muchos más elementos en la malla de *NetGen*, sin embargo si obtenemos el número de elementos de los que está compuesta la malla en cada uno de los casos vemos que no es así.

Al mallar el cilindro con *GiD*® se obtienen 1285 tetraedros mientras que si la malla se realiza con *NetGen* conseguimos que el valor baje hasta los 571 tetraedros. El hecho de tener

más tetraedros supone que al estudiar la estructura mediante métodos numéricos se tengan más incógnitas y aumente el tiempo de procesado de los datos. En este caso los resultados son mejores si mallamos con *NetGen*, por lo tanto la malla que proporciona el módulo *GiDtoNet* es mejor que la proporcionada directamente con *GiD*®.

5.4.2. Tiempo en la realización del mallado.

Este parámetro de calidad de los mallados mide el tiempo que tardan ambas herramientas en generar la malla de la estructura bajo estudio, en este caso un cilindro.

GiD® tarda 779 ms en realizar el mallado de la estructura y *NetGen* 688 ms. Con lo que tenemos una diferencia de unos 91 ms entre *GiD*® y *NetGen*. El usuario no es capaz de percibir esta diferencia de tiempo, ya que ambos generadores crean la malla mucho más rápido que en los dos casos de prueba anteriores, sin ambargo, tenemos que admitir que en este sentido mejoramos los resultados utilizando *NetGen*. En el mallado de esta estructura quizá no se aprecie la diferencia, pero si este valor se escala en función del número de elementos, puede que cada vez se aprecie más y sea determinante para elegir entre un generador de mallado y otro.

5.4.3. Mínimo ángulo de los tetraedros.

Este parámetro mide el ángulo interior mínimo de los tetraedros que componen la malla. En las gráficas 5.29 y 5.30 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea menor de 20 grados tienen peor calidad, por lo tanto el que más elementos tenga por debajo de este valor proporcionará una malla con unas características peores.

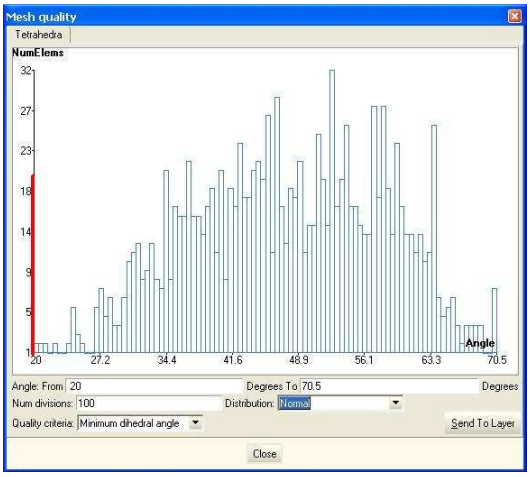


Figura 5.29: Tamaño mínimo del ángulo de los tetraedros con *GiD*.

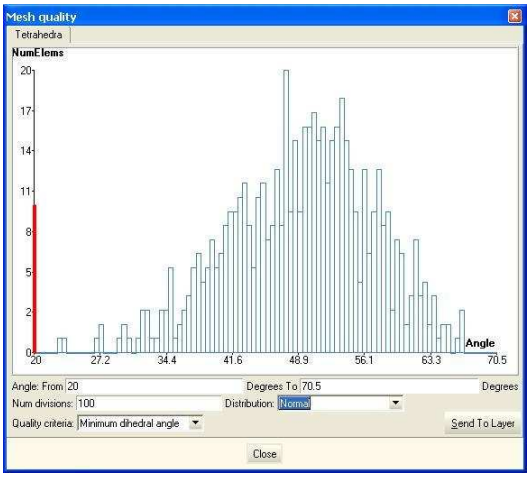


Figura 5.30: Tamaño mínimo del ángulo de los tetraedros con *NetGen*.

Como se observa en las gráficas 5.29 y 5.30, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros comienzan a perder calidad. En este caso no se obtienen tetraedros por debajo del umbral en ninguno de los casos, por lo tanto ambos mallados son buenos para trabajar con ellos. Sin embargo, en el caso de mallar con *GiD*® tenemos algunos tetraedros que se acercan al valor límite.

5.4.4. Máximo ángulo de los tetraedros.

Este parámetro mide el ángulo interior máximo de los tetraedros de una malla malla. En las gráficas 5.31 y 5.32 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea mayor de 120 grados tienen peor calidad, por lo tanto el que más elementos tenga por encima de este valor proporcionará una malla con unas características peores.

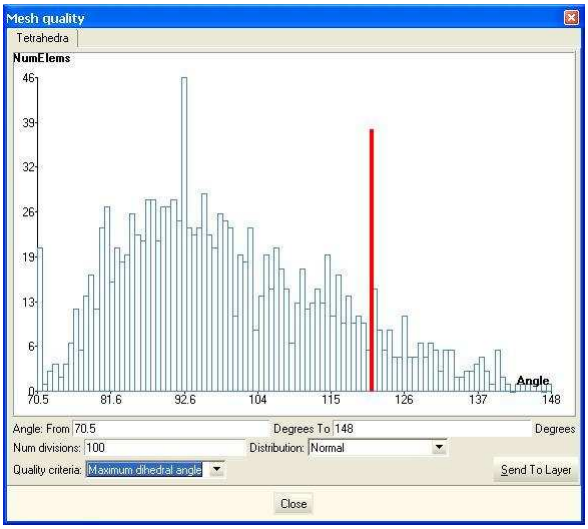


Figura 5.31: Tamaño máximo del ángulo de los tetraedros con *GiD*.

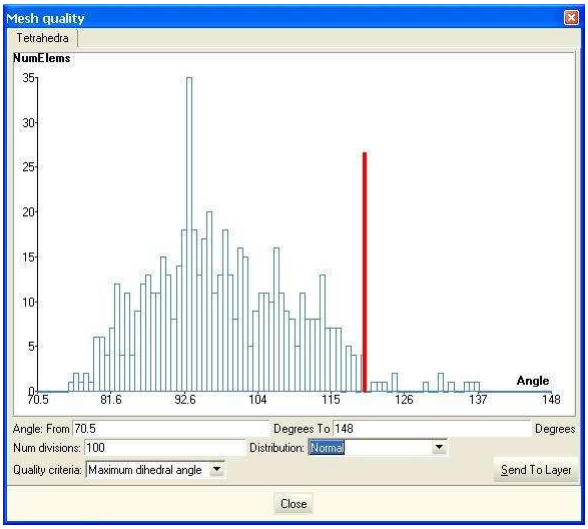


Figura 5.32: Tamaño máximo del ángulo de los tetraedros con *NetGen*.

Como se observa en las gráficas 5.31 y 5.32, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros pierden calidad. Si se cuenta el número de tetraedros que hay por encima de este umbral en uno y otro caso se obtienen los siguientes resultados:

- *GiD*®: 142 tetraedros. Que obteniendo el porcentaje respecto al número total de éstos que tiene la malla se obtiene un 11.05 %.
- *NetGen*: 14 tetraedros. Mediante la misma operación anterior se obtiene un 2.452 % repecto al número total de tetraedros del mallado.

Gracias a los resultados anteriores, se puede afirmar que la malla que genera *NetGen* tiene mejor calidad que la proporcionada por *GiD*®, ya que este último tiene un mayor porcentaje de tetraedros de peor calidad en su malla.

5.4.5. Forma de los tetraedros.

En las gráficas 5.33 y 5.34 están representados los resultados obtenidos para el mallado con *GiD*® y *NetGen*. Este parámetro de calidad se encarga de medir la relación que existe entre los tetraedros de la malla y el de referencia. Cuanto más se parezca un tetraedro al de referencia más cercano a uno estará el valor medido en este caso.

Si nos fijamos en los resultados obtenidos con ambos generadores de mallado, se observa en el caso de la malla creada con *GiD*® los valores están comprendidos entre 0.577 y 1, mientras que para el caso de *NetGen* los valores se encuentran entre 0.6295 y 0.947. La mayoría de los elementos en el caso de *GiD*® tienen una semejanza con el tetraedro de referencia de 0.753, mientras que para el caso de *NetGen* ese valor es 0.717.

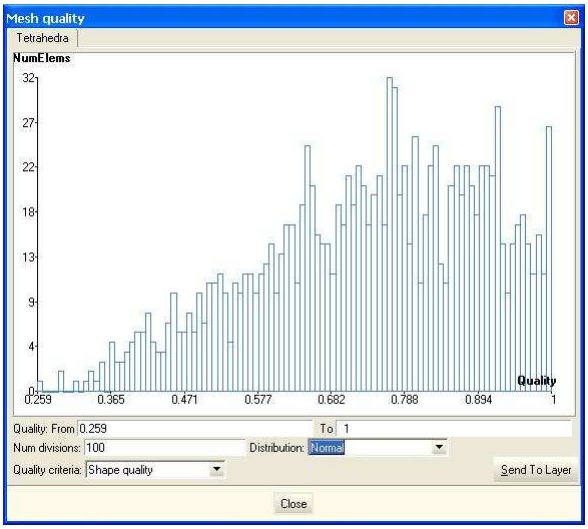


Figura 5.33: Forma de los tetraedros con *GiD*.

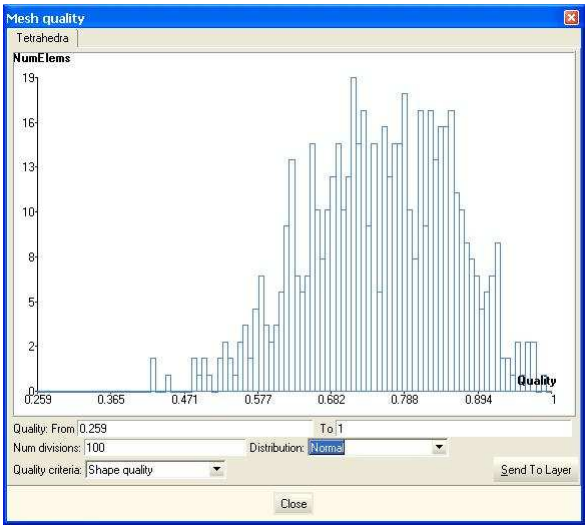


Figura 5.34: Forma de los tetraedros con *NetGen*.

En este caso, la forma que tienen la mayoría de los tetraedros en ambos mallados es muy parecida, siendo su valor ligeramente mayor en el caso de *GiD*®. Sin embargo, se considera que el mallado proporcionado con *NetGen* es mejor, a pesar de que GiD® tienen muchos de ellos exactamente iguales al de referencia, porque tiene menos tetraedros que se alejan de la forma de éste, manteniéndose su forma dentro de un rango muy estrecho.

5.4.6. Volumen de los elementos.

Las gráficas 5.35 y 5.36 recogen el volumen de cada uno de los tetraedros que forman la malla en ambos casos. Esta medida nos permitirá demostrar si tiene sentido el parámetro medido anteriormente, el número de elementos. La relación entre ambos parámetros es inversamente proporcional.

Si mallamos con *GiD*®, los valores de la gráfica están comprendidos entre 0.0.00595 y 0.00676, y en el caso de mallar con *NetGen* entre 0.00759 y 0.0138. Además, en el caso de *GiD*®, la mayoría de sus elementos tienen un volumen con un valor en torno a 0.00634, mientras que con *NetGen* se encuentran alrededor de 0.01256.

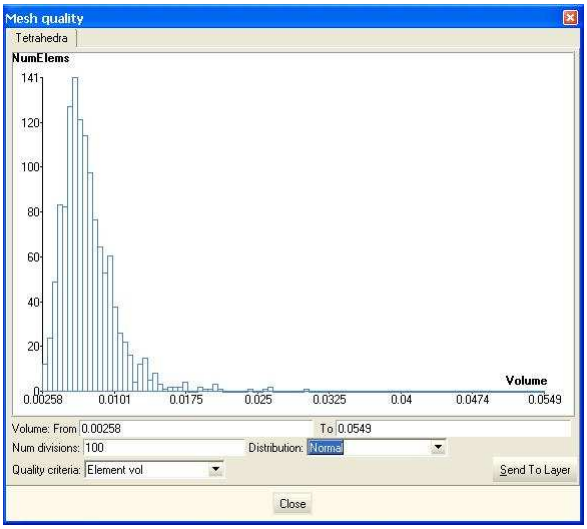


Figura 5.35: Volumen de los elementos con *GiD*.

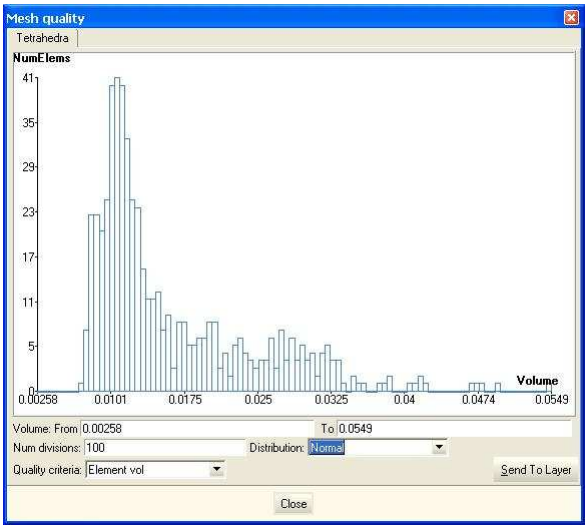


Figura 5.36: Volumen de los elementos con *NetGen*.

Tras estudiar los resultados obtenidos mediante las gráficas anteriores, podemos afirmar que se demuestra la relación entre los dos parámentros mencionados. A mayor número de elementos en la malla, menor volumen tendrán.

5.4.7. Conclusiones.

Tras analizar todos los parámetros de calidad del mallado, podemos afirmar que *GiDtoNet* proporciona un mallado al usuario (el mallado de *NetGen*) con mayor calidad que el proporcionado directamente con *GiD*®. Por lo tanto se le recomienda utilizar este módulo para hacer mallados con características similares.

A continuación se muestra una tabla en la que se recogen los resultados obtenidos para cada uno de los estudios de calidad realizados para que el usuario pueda tener una forma rápida de consultar los valores más importantes de cada una de las medidas realizadas.

	GiD®	**NetGen**
Número de tetraedros	1285	571
Tiempo generación mallado (ms)	779	688
Ángulo mínimo < 20 grados	0 tetraedros	0 tetraedros
% respecto al total	0 %	0 %
Ángulo máximos > 120 grados	142 tetraedros	14 tetraedros
% respecto al total	11.05 %	2.452 %
Forma de los tetraedros	0.753	0.717
Volumen de los elementos	0.00634	0.01256

5.5. Mallado de un tronco de cono.

Esta prueba consiste en la realización de un cono, con radio de la base 1 y altura 3, y su posterior mallado con *GiD®* y *NetGen*. La geometría del problema se puede observar en las figuras 5.37 y 5.38. La peculiaridad de esta estructura reside en que se representa de dos maneras

diferentes en función del generador de mallado que se utilice, ya que en *NetGen* se convierte en un tronco de cono, con el mismo radio de base y altura 2. Debido a esta representación, el estudio o comparación que se realizará a continuación no permitirá obtener unas conclusiones claras al igual que se ha hecho en las pruebas anteriores. Símplemente se ha querido estudiar este tipo de estructuras para observar el comportamiento de los dos generadores de mallado que estamos analizando.

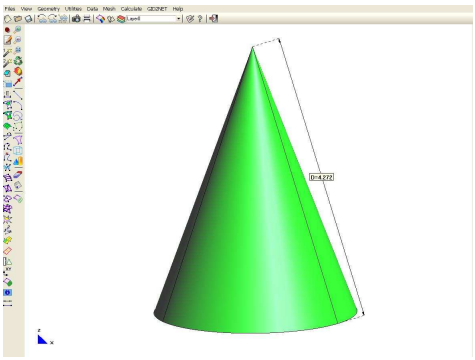


Figura 5.37: Estructura de un cono con *GiD*.

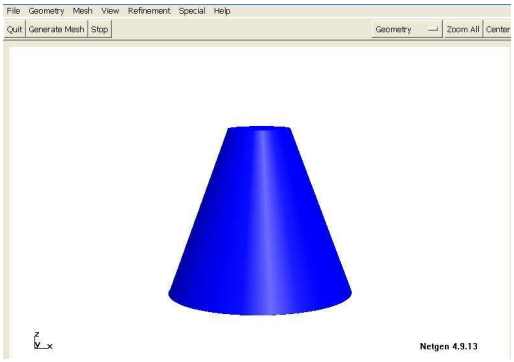


Figura 5.38: Estructura de un cono con *NetGen*.

En las figuras 5.39 y 5.40 se observan los mallados realizados con ambos generadores de mallado. En este caso se ha tomado como tamaño de mallado 0.6 y un grading de 0.3. El resto de parámetros permanecen constantes.

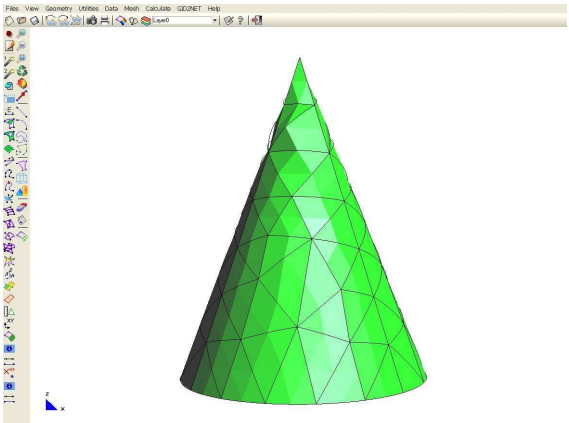


Figura 5.39: Mallado de un cono con *GiD*.

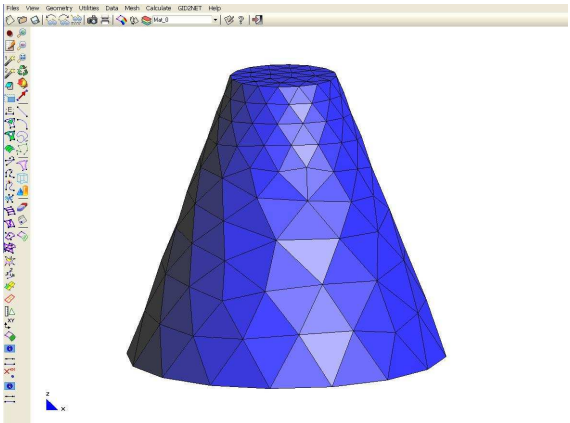


Figura 5.40: Mallado de un cono con *NetGen*.

5.5.1. Número de elementos en el mallado.

A simple vista se puede apreciar que para esta estructura, al contrario que en algunas de las estructuras anteriores, el mallado superficial en ambos casos no se parece. Esto se debe principalmente a que, al tener el mismo tamaño de malla y tener estructuras diferentes, uno generará elementos mayores que otro, ya que el volumen de la estructura en el caso de *GiD*® es mayor que en *NetGen*.

Al mallar el cono con *GiD*® se obtienen 338 tetraedros mientras que si mallamos el tronco de cono mediante *NetGen* se obtienen 522. El hecho de tener más tetraedros supone que al estudiar la estructura mediante métodos numéricos se tengan más incógnitas y aumente el tiempo de procesado de los datos. A diferencia de los casos anteriores, los resultados son mejores si mallamos con *GiD*®.

5.5.2. Tiempo en la realización del mallado.

Como se comentó en la introducción del presente capítulo y en cada una de las pruebas realizadas anteriormente, mediante este parámetro se mide el tiempo que tardan ambas herramientas en generar la malla de la estructura bajo estudio.

En este caso, para mallar un cono, hemos obtenido que *GiD*® tarda en realizar la malla 975 ms y *NetGen* 574 ms. Se observa que se tarda casi la mitad de tiempo en realizar la malla con *NetGen*, con lo que en este sentido se elegiría este generador de mallado para mallar la estructura.

5.5.3. Mínimo ángulo de los tetraedros.

Este parámetro mide el ángulo interior mínimo de los tetraedros que componen la malla. En las gráficas 5.41 y 5.42 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea menor de 20 grados tienen peor calidad, por lo tanto el que más elementos tenga por debajo de este valor proporcionará una malla con unas características peores.

Como se observa en las gráficas 5.41 y 5.42, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros son peores. Si se cuenta el número de tetraedros que hay por encima de este umbral en uno y otro caso se obtienen los siguientes resultados:

- *GiD*®: 1 tetraedro. Mediante la misma operación anterior se obtiene un 0.2 % repecto al número total de tetraedros del mallado.
- *NetGen*: 0 tetraedros. No tiene ninguno que esté por debajo del umbral.

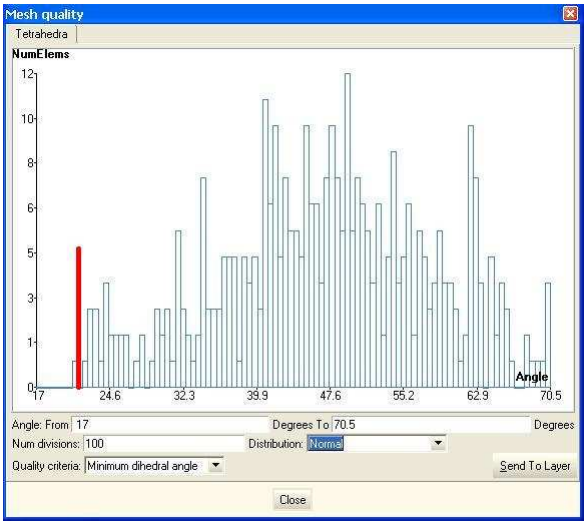


Figura 5.41: Tamaño mínimo del ángulo de los tetraedros con *GiD*.

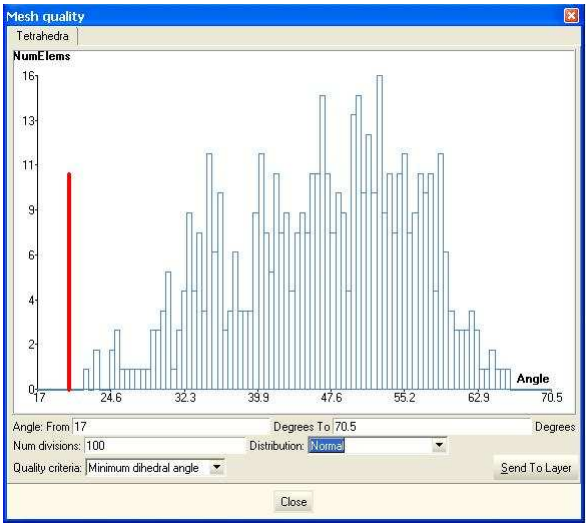


Figura 5.42: Tamaño mínimo del ángulo de los tetraedros con *NetGen*.

Una vez estudiados los valores numéricos recogidos, se comprueba que en este caso, los resultados no son concluyentes, ya que hay muy poca diferencia numérica entre los dos casos.

5.5.4. Máximo ángulo de los tetraedros.

Este parámetro mide el ángulo interior máximo de los tetraedros de una malla malla. En las gráficas 5.43 y 5.44 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea mayor de 120 grados tienen peor calidad, por lo tanto el que más elementos tenga por encima de este valor proporcionará una malla con unas características peores.

Como se observa en las gráficas 5.43 y 5.44, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros comienzan a perder calidad. Si se cuenta el número de tetraedros que hay por encima de este umbral en uno y otro caso se obtienen los siguientes resultados:

- *GiD*®: 47 tetraedros. Que obteniendo el porcentaje respecto al número total de éstos que tiene la malla se obtiene un 13.91 %.
- *NetGen*: 50 tetraedros. Mediante la misma operación anterior se obtiene un 9.06 % repecto al número total de tetraedros del mallado.

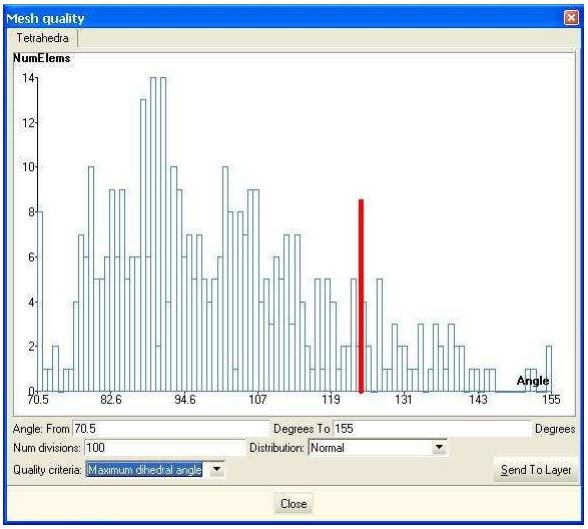


Figura 5.43: Tamaño máximo del ángulo de los tetraedros con *GiD*.

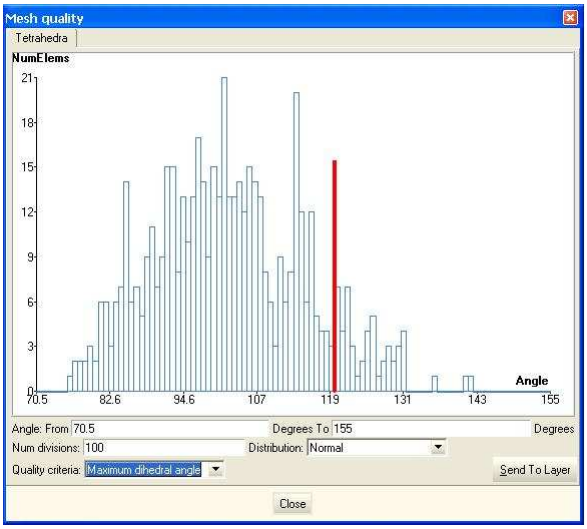


Figura 5.44: Tamaño máximo del ángulo de los tetraedros con *NetGen*.

Gracias a los resultados anteriores, se puede afirmar que la malla que genera *NetGen* tiene mejor calidad que la proporcionada por *GiD*®, ya que este último tiene un mayor porcentaje de tetraedros de peor calidad en su malla.

5.5.5. Forma de los tetraedros.

En las gráficas 5.45 y 5.46 están representados los resultados obtenidos para el mallado con *GiD*® y *NetGen*. Este parámetro de calidad se encarga de medir la relación que existe entre los tetraedros de la malla y el de referencia. Cuanto más se parezca un tetraedro al de referencia más cercano a uno estará el valor medido en este caso.

GiD® tiene la mayoría de sus tetraedros con una forma comprendida entre 0.523 y 1, mientras que para el caso de *NetGen* se encuentran entre 0.542 y 0.82. La mayoría de los elementos en el caso de *GiD*® tienen una semejanza con el tetraedro de referencia de 0.8215, mientras que para el caso de *NetGen* ese valor es 0.802.

Podemos decir que con *NetGen* se obtienen mejores resultados que con *GiD*®, ya que sus tetraedros, aunque no son exactamente iguales al de referencia, mantienen una forma casi constante dentro de un rango, mientras que *GiD*® tiene muchos de ellos con una forma que dista mucho de la del tetraedro de referencia.

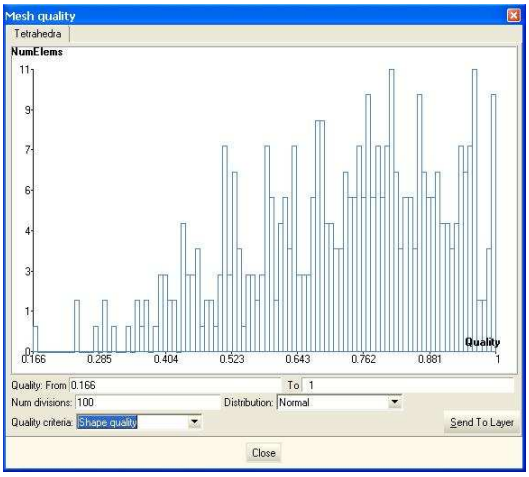


Figura 5.45: Forma de los tetraedros con *GiD*.

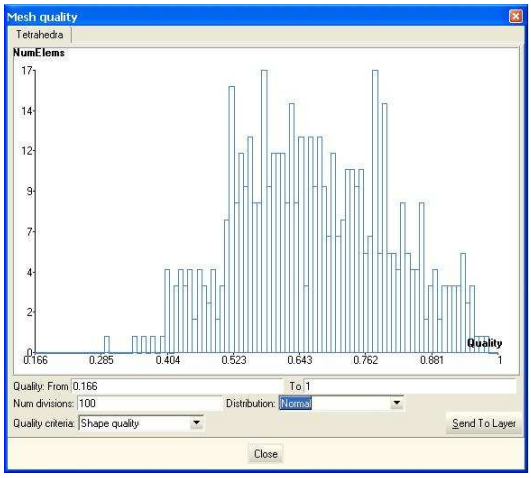


Figura 5.46: Forma de los tetraedros con *NetGen*.

5.5.6. Volumen de los elementos.

Las gráficas 5.47 y 5.48 recogen el volumen que tienen cada uno de los elementos de la malla llevada a cabo por los dos generadores de mallado bajo estudio. Este parámetro tiene una relación inversa con el número de elementos de los que está compuesta la malla. A más tetraedros, menor volumen tendrán cada uno de ellos.

Si mallamos con *GiD*®, los valores de la gráfica están comprendidos entre 0.0164 y 0.032, y en el caso de mallar con *NetGen* entre 0.000744 y 0.0112. Además, en el caso de *GiD*®, la mayoría de sus elementos tienen un volumen con un valor en torno a 0.0216, mientras que con *NetGen* se encuentran alrededor de 0.000774.

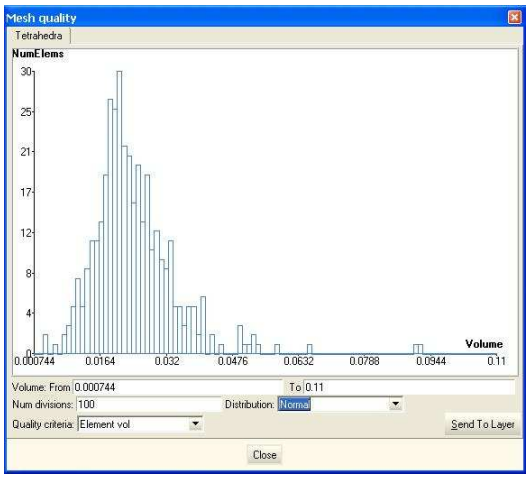


Figura 5.47: Volumen de los elementos con *GiD*.

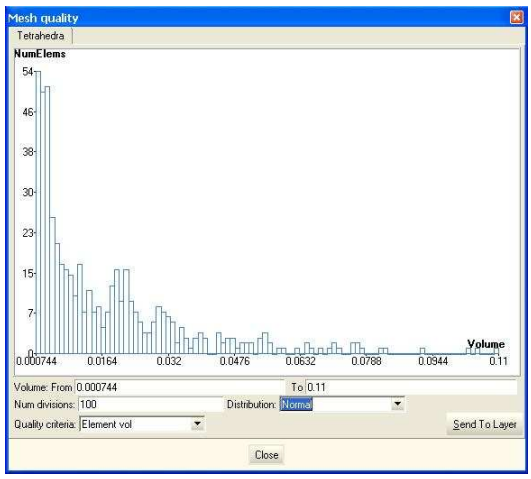


Figura 5.48: Volumen de los elementos con *NetGen*.

Los resultados de las gráficas anteriores demuestran que se cumple la relación entre los dos parámentros mencionados. En este caso, al tener más elementos la malla generada con *NetGen*, sus tetraedros tendrán menor volumen.

5.5.7. Conclusiones.

Como ya se ha comentado, no se pueden comparar mallados de estructuras que se representan de forma diferente. Con el estudio anterior, se pretendía medir los parámetros de calidad de la malla creada con *NetGen* para un tronco de cono frente a la que proporciona *GiD*®, pero debido a la forma que tiene cada generador de mallado de representar esta estructura, dicha comparación es difícil de realizar. No obstante, se muestra una tabla que recoge los resultados obtenidos para cada uno de los estudios de calidad realizados.

	GiD®	**NetGen**
Número de tetraedros	338	552
Tiempo generación mallado (ms)	975	574
Ángulo mínimo < 20 grados	1 tetraedro	0 tetraedros
% respecto al total	0.2 %	0 %
Ángulo máximos > 120 grados	47 tetraedros	50 tetraedros
% respecto al total	13.91 %	9.06 %
Forma de los tetraedros	0.8215	0.802
Volumen de los elementos	0.0216	0.000774

5.6. Mallado de una antena de bocina.

En este proyecto se va a analizar el comportamiento de los dos generadores de mallado bajo estudio frente a una estructura compuesta por dos entidades geométricas. Se va a proceder a la creación y el mallado de una antena de tipo bocina, veáse figura 5.50. Hay que resaltar que esta estructura está compuesta por un cilindro y un cono, por lo que hay que recordar que el cono presenta una estructura diferente en función del generador de mallado utilizado.

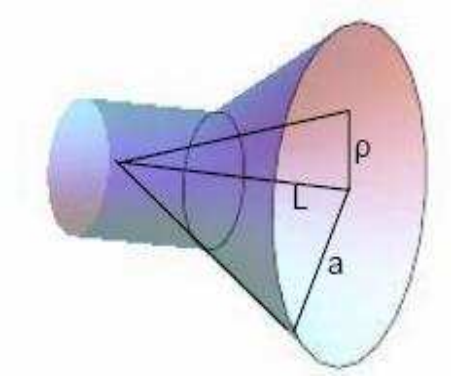


Figura 5.49: Ejemplo de una bocina.

La estructura que finalmente se obtiene representada en *GiD*® y *NetGen* se encuentra recogida en las figuras 5.50 y 5.51. En ellas podemos ver las dimensiones de la antena diseñada.

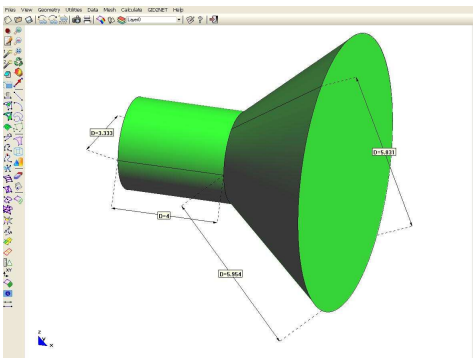


Figura 5.50: Bocina representada en *GiD*.

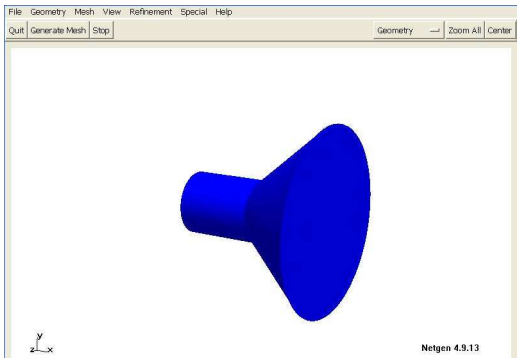


Figura 5.51: Bocina representada en *NetGen*.

En las figuras 5.52 y 5.53 se observan los mallados obtenidos con ambos generadores de mallado. En este caso, se ha utiliza un tamaño de malla de 0.6 y un grading de 0.3. El resto de parámetros son los mismos que en el resto de pruebas desarrolladas anteriormente.

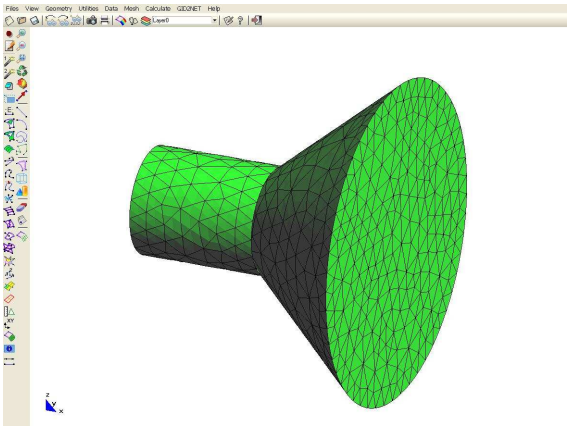


Figura 5.52: Mallado de una bocina con *GiD*.

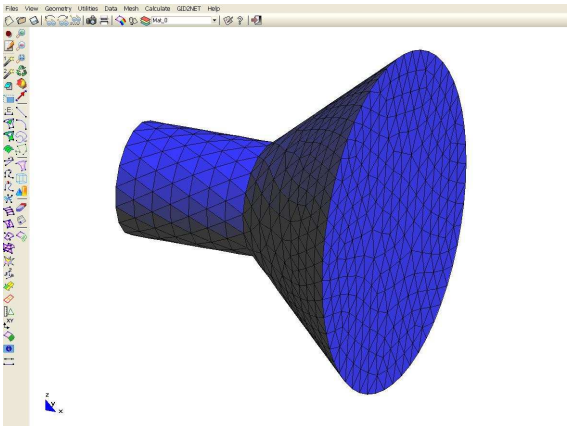


Figura 5.53: Mallado de una bocina con *NetGen*.

5.6.1. Número de elementos en el mallado.

A simple vista se puede apreciar que para esta estructura el mallado superficial que se ha creado en ambos casos es muy parecido en las dos partes que componen la estructura final.

En el mallado de la antena tipo bocina con *GiD*® se obtienen 8560 tetraedros mientras que si mallamos el tronco de cono mediante *NetGen* se obtienen 3119, casi una tercera parte respecto al primero. El hecho de tener más tetraedros supone que al estudiar la estructura mediante métodos numéricos se tengan más incógnitas y aumente el tiempo de procesado de los datos. Basándonos en este estudio podemos afirmar que la malla que proporciona el módulo *GiDtoNet* es mejor.

5.6.2. Tiempo en la realización del mallado.

Este parámetro se encarga de medir el tiempo que tardan ambas herramientas en realizar la malla de la estructura bajo estudio, que en este caso es una antena de tipo bocina.

Para este caso en concreto hemos obtenido que *GiD*® tarda en realizar la malla 2.261 s y *NetGen* 2.734 s. Tenemos una diferencia de unos 473 ms entre *GiD*® y *NetGen*. Esta diferencia apenas es apreciable cuando hablamos en términos de segundos, ya que medio segundo arriba o abajo no es relevante en la realización de las mallas. Sin embargo, puede que escalando este valor en función del número de elementos, si el proyecto aumentase en dimensión, sí que sea importante la diferencia de tiempos que se obtenga.

5.6.3. Mínimo ángulo de los tetraedros.

Este parámetro mide el ángulo interior mínimo de los tetraedros que componen la malla. En las gráficas 5.54 y 5.55 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea menor de 20 grados tienen peor calidad, por lo tanto el que más elementos tenga por debajo de este valor proporcionará una malla con unas características peores.

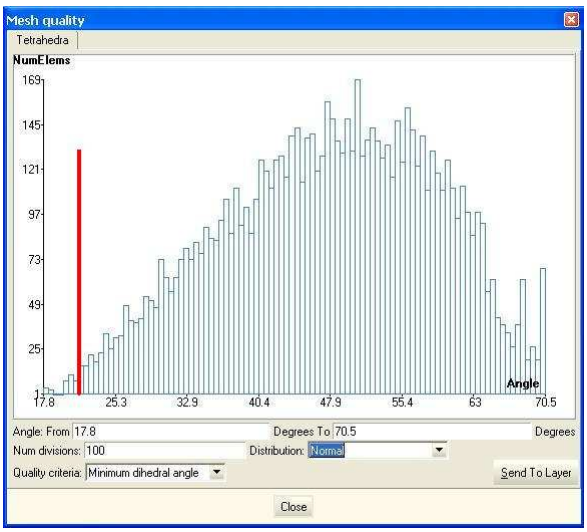


Figura 5.54: Tamaño mínimo del ángulo de los tetraedros con *GiD*.

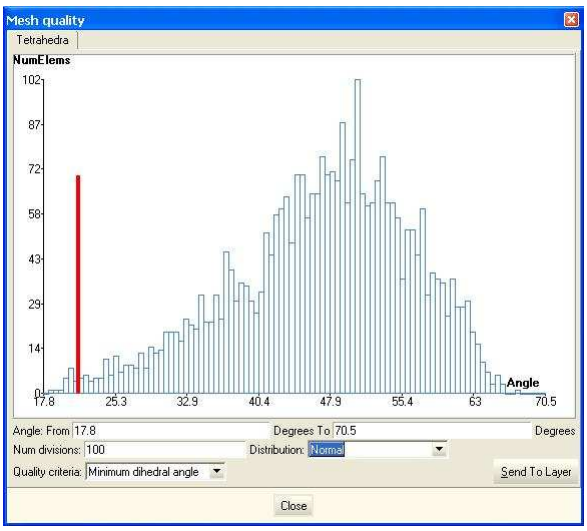


Figura 5.55: Tamaño mínimo del ángulo de los tetraedros de los elementos con *NetGen*.

Como se observa en las gráficas 5.54 y 5.55, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros son peores. Si se cuenta el número de tetraedros que hay por encima de este umbral en uno y otro caso se obtienen los siguientes resultados:

- *GiD*®: 27 tetraedros. Que obteniendo el porcentaje respecto al número total de éstos que tiene la malla se obtiene un 0.3154 %.
- *NetGen*: 14 tetraedros. Mediante la misma operación anterior se obtiene un 0.449 % repecto al número total de tetraedros del mallado.

Por lo tanto, habiendo analizado los valores obtenidos vemos, que según los porcentajes hay más elementos de peor calidad en *NetGen* que en *GiD*®, aunque esa diferencia es mínima. Para este caso se recomienda el mallado con *GiD*®.

5.6.4. Máximo ángulo de los tetraedros.

Este parámetro mide el ángulo interior máximo de los tetraedros de una malla malla. En las gráficas 5.56 y 5.57 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea mayor de 120 grados tienen peor calidad, por lo tanto el que más elementos tenga por encima de este valor proporcionará una malla con unas características peores.

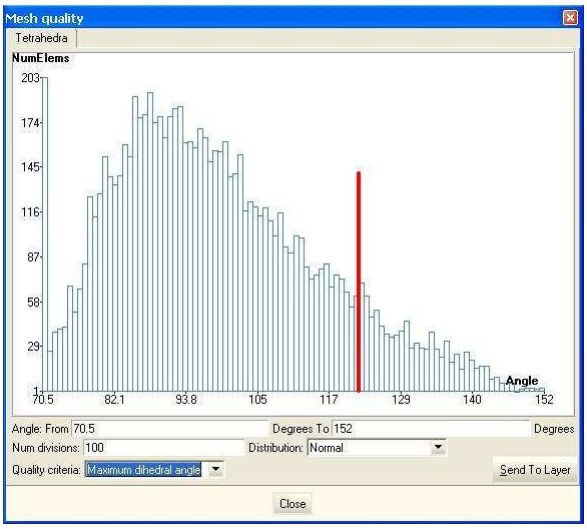


Figura 5.56: Tamaño máximo del ángulo de los tetraedros con *GiD*.

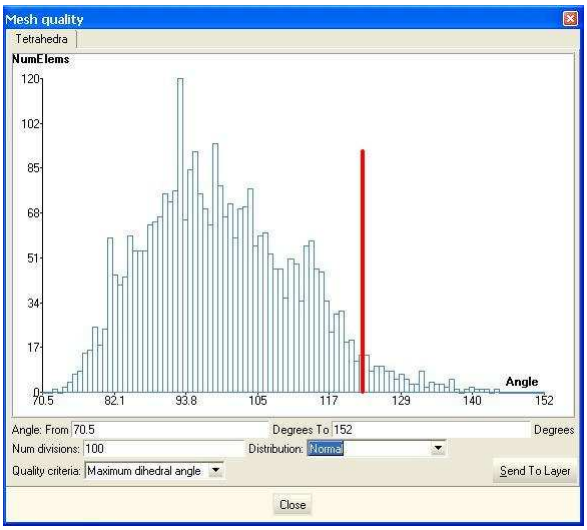


Figura 5.57: Tamaño máximo del ángulo de los tetraedros con *NetGen*.

Como se observa en las gráficas 5.56 y 5.57, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros pierden calidad. Si se cuenta el número de tetraedros que hay por encima de este umbral en uno y otro caso se obtienen los siguientes resultados:

- *GiD*®: 924 tetraedros. Que obteniendo el porcentaje respecto al número total de éstos que tiene la malla se obtiene un 10.79 %.
- *NetGen*: 41 tetraedros. Mediante la misma operación anterior se obtiene un 1.31 % repecto al número total de tetraedros del mallado.

Gracias a los resultados anteriores, se puede afirmar que la malla que genera *NetGen* en este caso tiene mejor calidad que la proporcionada por *GiD*®, ya que este último tiene un mayor porcentaje de tetraedros de peor calidad en su malla.

5.6.5. Forma de los tetraedros.

En las gráficas 5.58 y 5.59 están representados los resultados obtenidos para el mallado con *GiD*® y *NetGen*. Este parámetro de calidad se encarga de medir la relación que existe entre los tetraedros de la malla y el referencia. Cuanto más se parezca un tetraedro al de referencia más cercano a uno estará el valor medido en este caso.

Si nos fijamos en los resultados obtenidos en las gráficas, se observa que para la mallada creada con *GiD*® los valores obtenidos para medir la forma de sus tetraedros están comprendidos entre 0.581 y 1, mientras que para el caso de *NetGen* estos valores se encuentran entre 0.6275 y 0.941. La mayoría de los elementos en el caso de *GiD*® tienen una semejanza con el tetraedro de referencia de 1, son prácticamente iguales, mientras que para el caso de *NetGen* ese valor es 0.784.

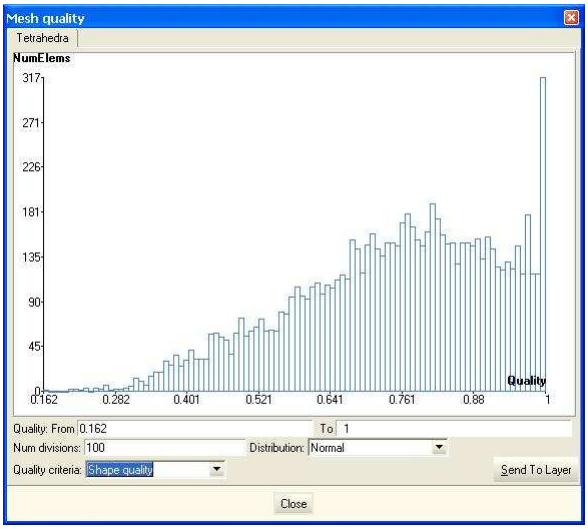


Figura 5.58: Forma de los tetraedros con *GiD*.

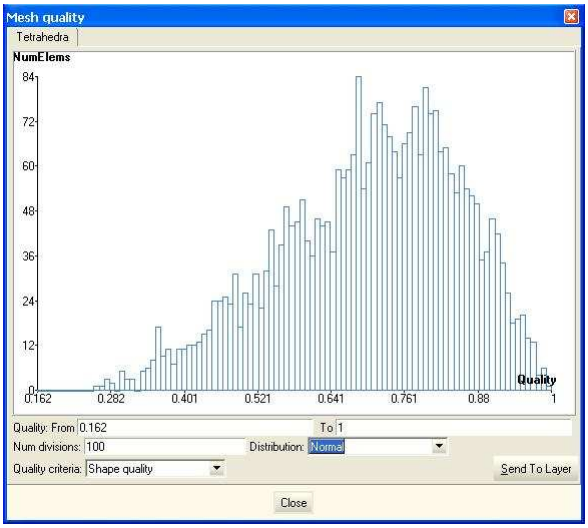


Figura 5.59: Forma de los tetraedros con *NetGen*.

Tras analizar los resultados proporcionados por las gráficas, podemos afirmar que con *NetGen* se obtienen mejores resultados que con *GiD*®, ya que de nuevo sus tetraedros, aunque no son exactamente iguales al tetraedro de referencia, mantienen una forma constante, mientras que *GiD*® tiene muchos de ellos en los que su forma dista mucho de la del tetraedro de referencia.

5.6.6. Volumen de los elementos.

Las gráficas 5.60 y 5.61 recogen el valor del volumen de los tetraedros que componen la malla de la antena de bocina en ambos casos. Este parámetro sirve para verficar que el número de elementos obtenido en el estudio anterior. Como ya se ha comentado anteriormente, la relación entre estos dos parámetros de calidad es inversamente proporcional, ya que a mayor número de elementos, menor volumen tendrán.

Si mallamos con *GiD*®, los valores de la gráfica están comprendidos entre 0.01056 y 0.0266, y en el caso de mallar con *NetGen* entre 0.0186 y 0.083. Además, en el caso de *GiD*®, la mayoría de sus elementos tienen un volumen con un valor en torno a 0.0186, mientras que con *NetGen* se encuentran alrededor de 0.0454.

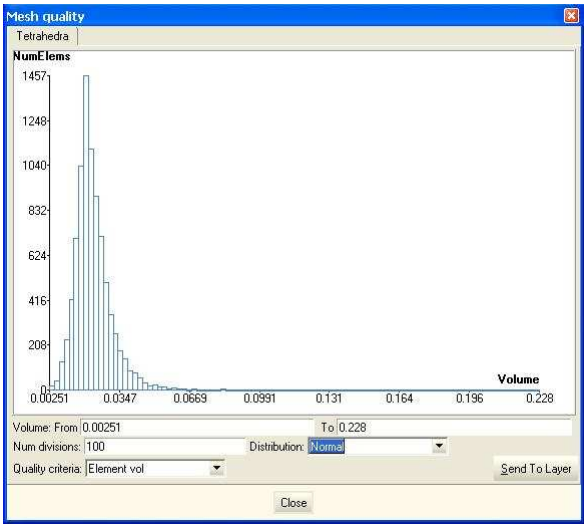


Figura 5.60: Volumen de los elementos con *GiD*.

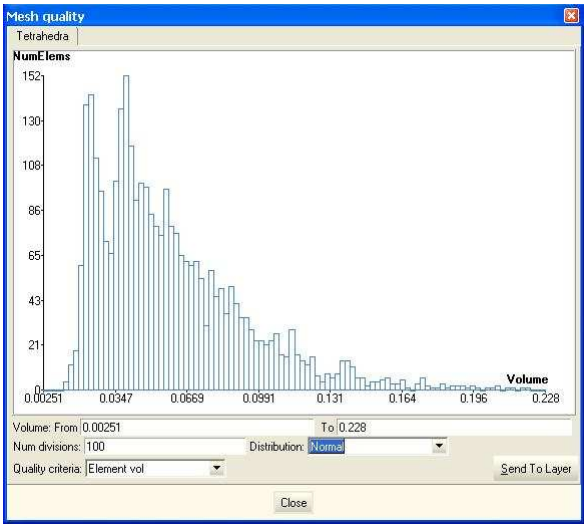


Figura 5.61: Volumen de los elementos con *NetGen*.

Tras estudiar los resultados obtenidos con ambos generadores de mallado para este parámetro de calidad se ha comprobado que a mayor número de elementos en la malla, menor volumen tendrán. En este caso, al tener menos elementos la malla generada con *NetGen*, sus tetraedros tendrán mayor volumen.

5.6.7. Conclusiones.

A continuación se muestra una tabla en la que se recogen los resultados obtenidos para cada uno de los estudios de calidad realizados. A pesar de que dos de los estudios realizados en este caso salen ligeramente peores en el caso de mallar con *NetGen*, las diferencias numéricas encontradas no son muy significativas. Por lo tanto, se sigue considerando mejor el mallado realizado mediante el módulo *GiDtoNet*. Aunque se deja libertad al usuario para que sea él quien decida.

	GiD®	**NetGen**
Número de tetraedros	8560	3119
Tiempo generación mallado (s)	2.261	2.734
Ángulo mínimo < 20 grados	27 tetraedros	14 tetraedros
% respecto al total	0.3154 %	0.449 %
Ángulo máximos > 120 grados	924 tetraedros	41 tetraedros
% respecto al total	10.79 %	1.31 %
Forma de los tetraedros	1	0.784
Volumen de los elementos	0.0186	0.0454

5.7. Mallado de un proyecto completo. Hércules C-130.

Para poder hacer una verdadera comparación entre las propiedades de los mallados generados por las dos herramientas bajo estudio se necesita crear un proyecto real en el que el usuario interaccione más a fondo con el módulo. Para ello se ha elegido la estructura que presenta un avión Hércules C-130, veáse figura 5.62. Dicha estructura se creará a partir de entidades geométricas simples y diversas operaciones entre ellas intentando que el modelo quede lo más realista posible, pero el lector debe comprender la complejidad del proyecto y las limitaciones en las entidades que se pueden representar.

Figura 5.62: Hércules C-130.

Se comenzará creando las entidades geométricas que darán forma a la estructura del avión que hemos tomado como referencia. Como se aprecia en la figura 5.63, no se puede representar de forma exacta un Hércules C-130 debido a las limitaciones que presenta el módulo, pero el esquema volumétrico se asemeja bastante bien a la forma de un avión de este tipo. En esta figura se puede apreciar además las dimensiones de cada una de las partes del avión.

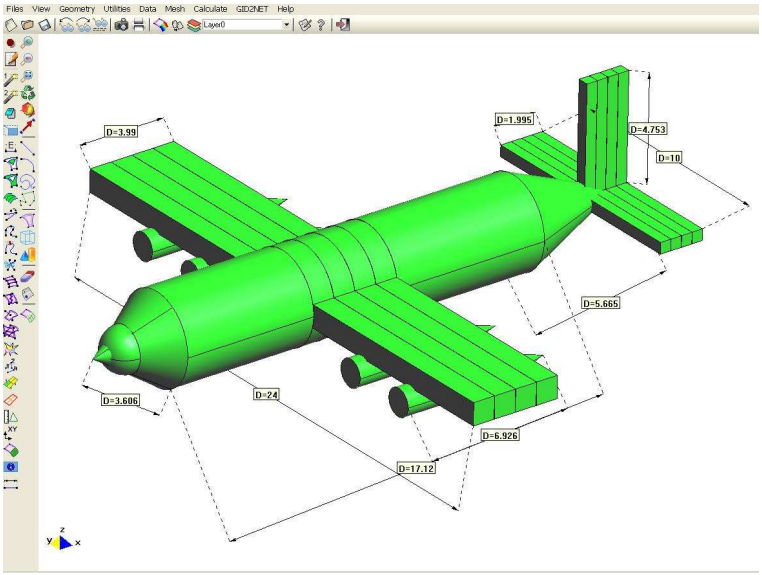


Figura 5.63: Hércules C-130 creado en *GiD* a partir de entidades geométricas simples y operaciones entre ellas.

Una vez generado por completo el diseño de la estructura bajo estudio, se creará el fichero *GiDtoNet.geo*, que contendrá las primitivas de todas las entidades geométricas que están presentes en el proyecto. En la figura 5.64 encontramos la representación de la estructura pero en la interfaz del generador de mallado *NetGen*. Así se comprueba funcionamiento del módulo, ya que permite crear el fichero con la información de geometría del proyecto y *NetGen* la genera correctamente.

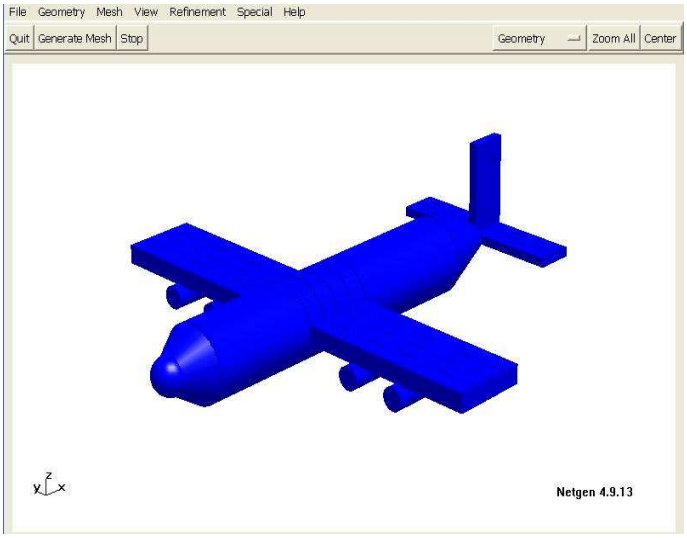


Figura 5.64: Representación de la estructura de la figura 5.63 en *NetGen*.

A continuación creamos el mallado de la estructura. Los parámetros de mallado que se han utilizado son tamaño de malla 0.9 y grading 0.3, el resto de ellos son los mismos que se han utilizado en el resto de pruebas. Como se puede observar, en las figuras 5.65 y 5.66 están representados los mallados obtenidos con *GiD*® y *NetGen* respectivamente. A simple vista vemos que los dos mallados son muy similares, al menos la malla de la superficie exterior.

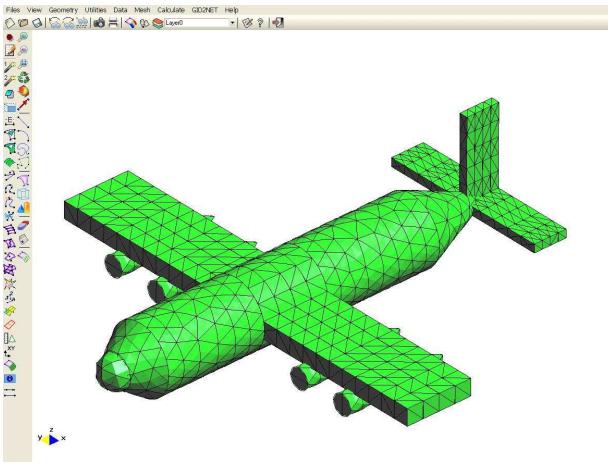


Figura 5.65: Mallado con *GiD*.

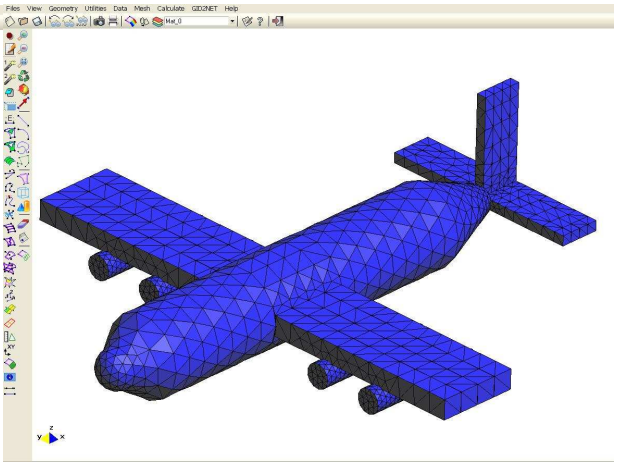


Figura 5.66: Mallado con *NetGen*.

En las figuras 5.67 y 5.67 se pueden observar dos de las partes del avión, el ala y los motores. Si nos centramos primero en el ala podemos apreciar que el mallado superficial que han creado ambas estructuras es muy similar, pero si nos fijamos en los motores se comprueba que el generador de mallado *NetGen* se adapta mucho mejor a la forma curva de su estructura. Esto hace que malle los motores con un mayor número de tetraedros consiguiendo así una forma prácticamente perfecta. Gráficamente podemos afirmar que la malla con *NetGen* se adapta mejor a la estructura que la creada por *GiD*®.

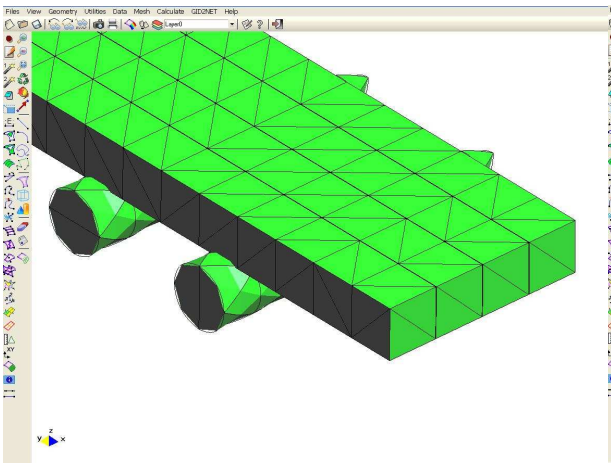


Figura 5.67: Mallado del ala con *GiD*.

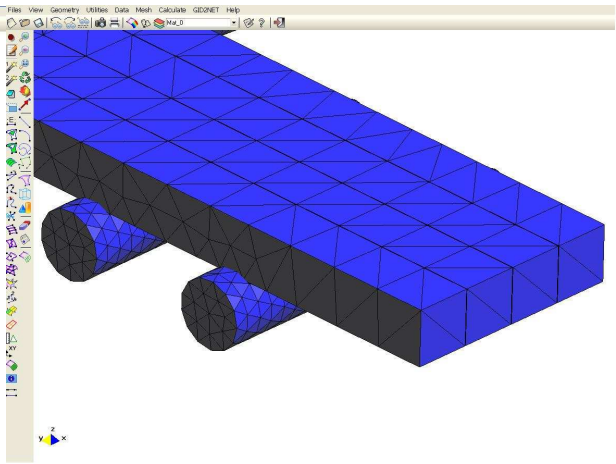


Figura 5.68: Mallado del ala con *NetGen*.

La siguiente parte de la estructura que se va a analizar es el morro del avión. Como se puede observar en las figuras 5.69 y 5.70, en las zonas en las que la estructura presenta superficies curvas, el generador de mallado *NetGen* se aproxima mejor a ellas. Nos podemos fijar en el número de elementos que presenta la esfera que contiene el morro, hay más tetraedros en la malla creada por *NetGen* que en la creada por *GiD*®.

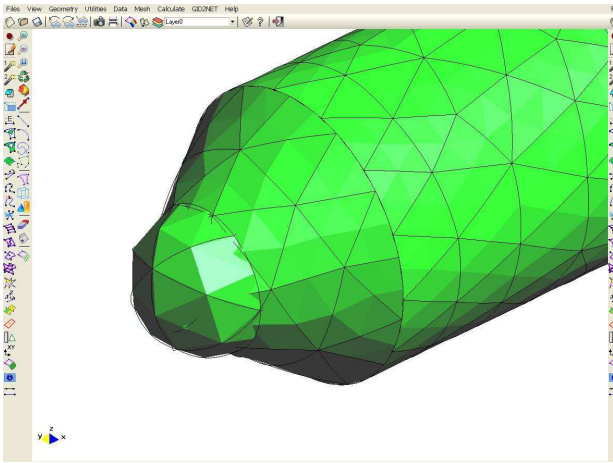


Figura 5.69: Mallado del morro con *GiD*.

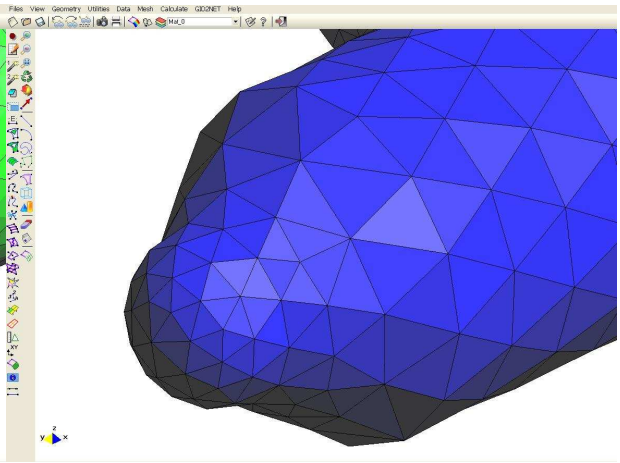


Figura 5.70: Mallado del morro con *NetGen*.

El último de los elementos que vamos a analizar con más detalle es la cola del avión, véase figuras 5.71 y 5.72. Ésta está compuesta por una serie de hexaedros que le confieren una robustez mayor y que hace que el mallado superficial con los dos generadores de mallado sea prácticamente el mismo. Sin embargo, en la zona cónica podemos identificar mayor densidad de tetraedros en el caso de *NetGen* que en el de *GiD*®.

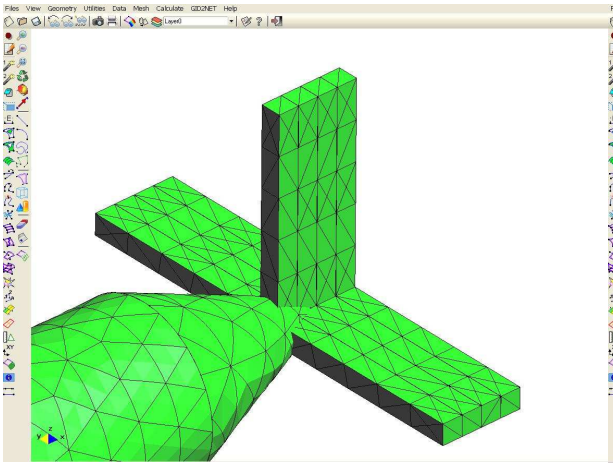


Figura 5.71: Mallado de la cola con *GiD*.

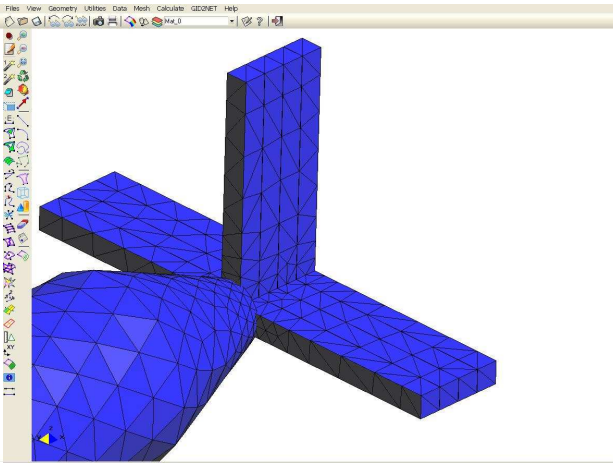


Figura 5.72: Mallado de la cola con *NetGen*.

5.7.1. Número de elementos del mallado.

En este caso se presenta el mallado de un proyecto que podría ser real. En él se realiza el mallado de una estructura compuesta por un gran número de entidades geométricas simples. Como ya se ha demostrado con los detalles de cada una de las partes del avión, el mallado con los dos generadores de mallado es muy similar excepto en aquellas partes que presentan superficies curvas o son elementos muy pequeños. En esos casos se adapta mejor la malla creada con *NetGen*.

Utilizando el generador de mallado *GiD*®,tenemos una malla formada por 5421 tetraedros y si mallamos la estructura con *NetGen* el número de tetraedros es de 10588, casi el doble que en el primer caso. Este aumento en el número de elementos de la malla con *NetGen* es debido a la gran adaptatividad que presenta en superficies curvas, ya que necesita crear una malla más fina para poder representar los elementos con mayor precisión.

El aumento del número de tetraedros de la malla creada con *NetGen* hará que al intentar resolver, por métodos numéricos, la estructura en cuestión tengamos un mayor número de incógnitas, aumentando el tiempo de procesado de las soluciones.

5.7.2. Tiempo en la realización del mallado.

Este parámetro de calidad, como ya se ha ido comentando en el resto de pruebas, mide el tiempo que tardan cada una de las herramientas en realizar el mallado del avión.

GiD® malla la estructura en 8.47 s y *NetGen* en 12.656 s. La diferencia de tiempos entre un generador de mallado y otro a la hora de realizar la malla es importante, ya que hay una diferencia de 4.186 s entre ellos. Si nos centramos en los resultados obtenidos gracias a este parámetro de medida de calidad, tenemos que decir que es mejor utilizar *GiD*® para llevar a cabo el mallado de la estructura.

5.7.3. Mínimo ángulo de los tetraedros.

Este parámetro mide el ángulo interior mínimo de los tetraedros que componen la malla. En las gráficas 5.73 y 5.74 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea menor de 20 grados tienen peor calidad, por lo tanto el que más elementos tenga por debajo de este valor proporcionará una malla con unas características peores.

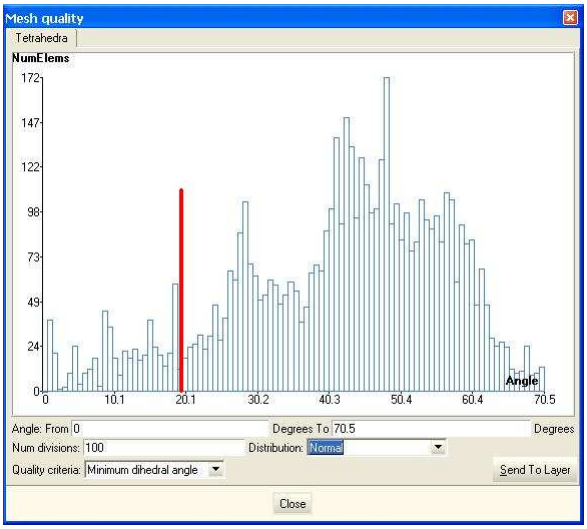


Figura 5.73: Tamaño mínimo del ángulo de los tetraedros en *GiD*.

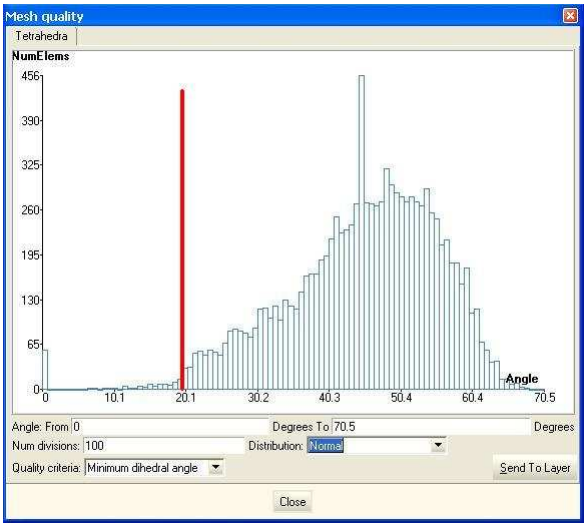


Figura 5.74: Tamaño mínimo del ángulo de los tetraedros en *NetGen*.

Como se observa en las gráficas 5.73 y 5.74, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros comienzan a ser peores. Si se cuenta el número de tetraedros que hay por encima de este umbral en uno y otro caso se obtienen los siguientes resultados:

- *GiD*®: 578 tetraedros. Que obteniendo el porcentaje respecto al número total de éstos que tiene la malla se obtiene un 10.66 %.
- *NetGen*: 122 tetraedros. Mediante la misma operación anterior se obtiene un 1.14 % repecto al número total de tetraedros del mallado.

Una vez estudiados los valores numéricos recogidos, se puede afirmar que la malla creada con *NetGen* presenta mejores características en este caso que la creada con *GiD*®.

5.7.4. Máximo ángulo de los tetraedros.

Este parámetro mide el ángulo interior máximo de los tetraedros de una malla malla. En las gráficas 5.75 y 5.76 encontramos representados los resultados obtenidos al realizar el mallado con las dos herramientas bajo estudio. En este caso el criterio establece que todos aquellos tetraedros cuyo ángulo interior sea mayor de 120 grados tienen peor calidad, por lo tanto el que más elementos tenga por encima de este valor proporcionará una malla con unas características peores.

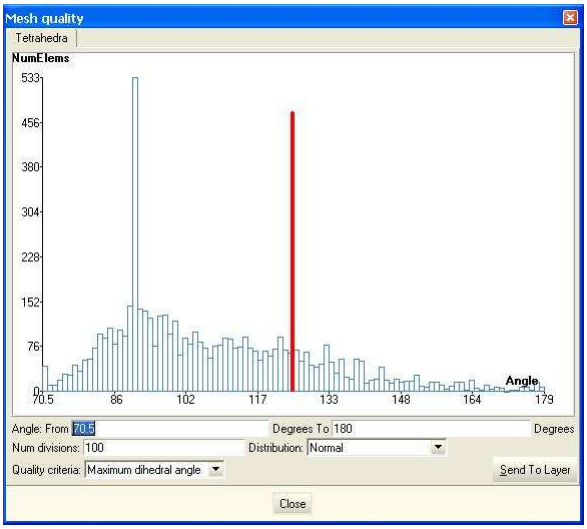


Figura 5.75: Tamaño máximo del ángulo de los tetraedros con *GiD*.

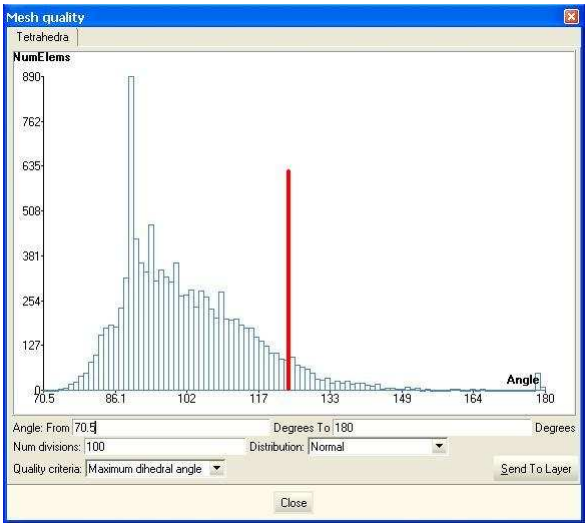


Figura 5.76: Tamaño máximo del ángulo de los tetraedros con *NetGen*.

Como se observa en las gráficas 5.75 y 5.76, mediante una línea en rojo se ha indicado el umbral en el que los tetraedros pierden calidad. Si se cuenta el número de tetraedros que hay por encima de este umbral en uno y otro caso se obtienen los siguientes resultados:

- *GiD*®: 1200 tetraedros. Que obteniendo el porcentaje respecto al número total de éstos que tiene la malla se obtiene un 22.14 %.
- *NetGen*: 929 tetraedros. Mediante la misma operación anterior se obtiene un 8.69 % repecto al número total de tetraedros del mallado.

Gracias a los resultados anteriores, se puede afirmar que la malla que genera *NetGen* tiene mejor calidad que la proporcionada por *GiD*®, ya que este último tiene un mayor porcentaje de tetraedros de peor calidad en su malla.

5.7.5. Forma de los tetraedros.

En las gráficas 5.33 y 5.34 están representados los resultados obtenidos para el mallado con *GiD*® y *NetGen*. Este parámetro de calidad se encarga de medir la relación que existe entre los tetraedros de la malla y el referencia. Cuanto más se parezca un tetraedro al de referencia más cercano a uno estará el valor medido en este caso.

Los resultados obtenidos indican que para la mallada creada con *GiD*® los valores para la forma de los tetraedros están comprendidos entre 0.153 y 1, mientras que para el caso de *NetGen* estos valores se encuentran entre 0.378 y 0.92. La mayoría de los elementos en el caso

de *GiD*® tienen una semejanza con el tetraedro de referencia de 0.623, mientras que para el caso de *NetGen* ese valor es 0.637.

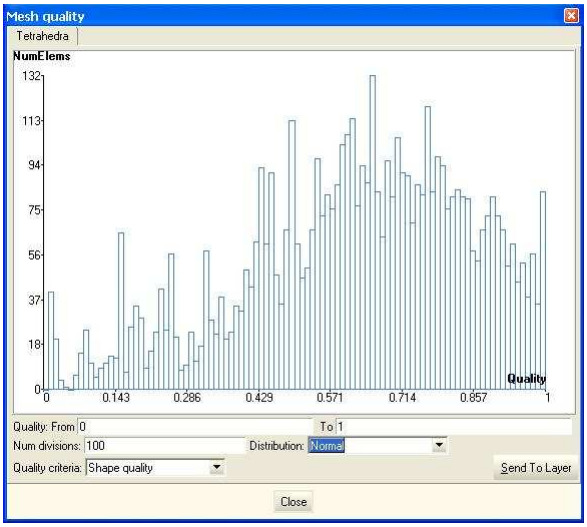


Figura 5.77: Forma de los tetraedros en *GiD*.

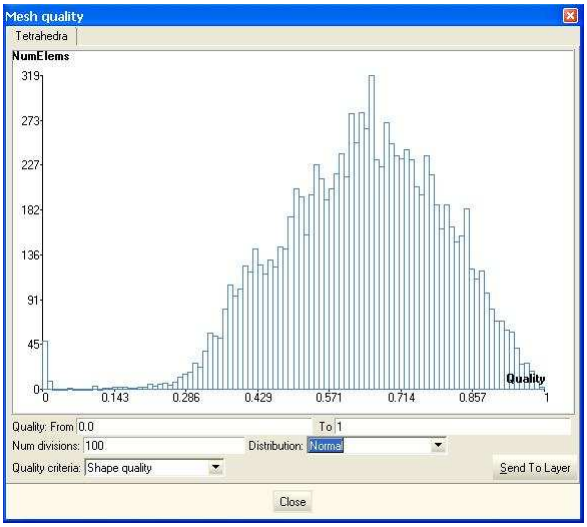


Figura 5.78: Forma de los tetraedros en *NetGen*.

En este caso los dos generadores de mallado tienen la mayoría de sus tetraedros con una forma en torno a 0.6 respecto al tetraedro de referencia. Como *NetGen* tiene muchos de sus tetraedros con una forma más o menos constante, dentro de un rango pequeño, y cercana al tetraedro de referencia, su mallado se considera mejor que el generado con *GiD*®, ya que éste último tiene muchos de sus tetraedros alejados de la forma que se toma como referencia.

5.7.6. Volumen de los elementos.

Las gráficas 5.79 y 5.80 recogen el volumen de cada uno de los elementos que forman la malla creada con cada uno de los generadores de mallado bajo estudio. Este parámetro servirá para vericar que el número de elementos obtenido en el parámetro anterior es correcto.

Si mallamos con *GiD*®, los valores de la gráfica están comprendidos entre 0 y 0.121, y en el caso de mallar con *NetGen* entre 0 y 0.0807. Además, en el caso de *GiD*®, la mayoría de sus elementos tienen un volumen con un valor en torno a 0.0807, mientras que con *NetGen* se encuentran alrededor de 0, es decir, un valor tan pequeño que con la escala de la que se dispone es difícil representarlo.

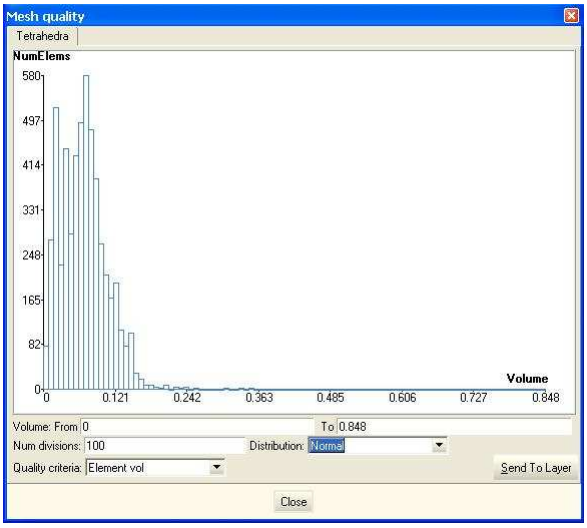


Figura 5.79: Volumen de los elementos en *GiD*.

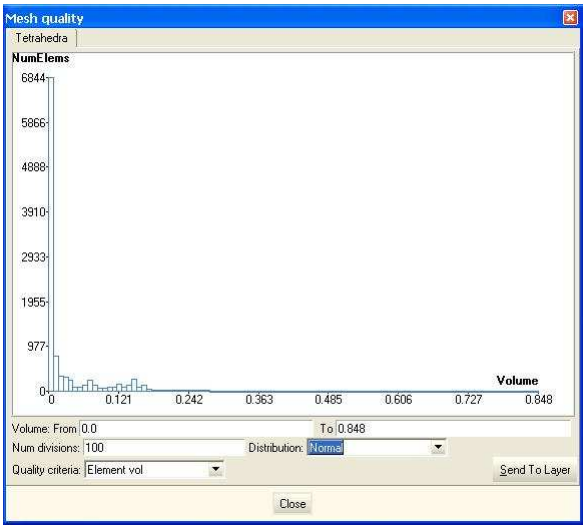


Figura 5.80: Volumen de los elementos en *NetGen*.

En este caso en particular, hemos comprobado que al medir el volumen de los elementos, tenemos que el tamaño de éstos en el caso de *GiD*® es mayor que en *NetGen*. Mediante este estudio podemos comprobar la validez del número de elementos que se ha recogido en uno de los parámetros anteriores. Cuantos más tetraedros compongan la malla, menor volumen tendrán cada uno de ellos.

5.7.7. Conclusiones.

Una vez que se han realizado las medidas de todos los parámetros de calidad que se van a estudiar, ya tenemos una base en la que fundamentar las conclusiones obtenidas sobre este proyecto en concreto.

Tras analidar los resultados obtenidos, se llega a la conclusión de que merece la pena tener un mayor número de elementos en la malla ya que proporciona una gran precisión en la representación de la estructura original, hemos podido observarlo gráficamente.

En cuanto al tiempo en la realización del mallado, hemos comprobado que se tardan cerca de 4 segundos más en mallar la estructura con *NetGen*, pero queda a elección del usuario priorizar entre los distintos parámetros estudiados a lo largo de esta sección para elegir entre un generador de mallado y otro para mallar su proyecto.

Se ha comprobado que el resto de parámetros estudiados para mallar una estructura compleja como el Hércules C-130, son mejores si se utiliza el módulo *GiDtoNet*. Con todo esto no se pretende afirmar rotundamente que un generador de mallado es mejor que otro, símplemente facilitarle al usuario la decisión que debe tomar a la hora de mallar sus proyectos.

A continuación se muestra una tabla en la que se recogen los resultados obtenidos para cada uno de los estudios de calidad realizados gracias a las herramientas proporcionadas por *GiD*®.

	GiD®	**NetGen**
Número de tetraedros	5421	10688
Tiempo generación mallado (s)	8.470	12.656
Ángulo mínimo < 20 grados	578 tetraedros	122 tetraedros
% respecto al total	10.66 %	1.14 %
Ángulo máximos > 120 grados	1200 tetraedros	929 tetraedros
% respecto al total	22.14 %	8.69 %
Forma de los tetraedros	0.623	0.637
Volumen de los elementos	0.0806	aprox. 0

Capítulo 6

Conclusiones y Futuras Líneas de Investigacion.

6.1. Conclusiones.

A lo largo de este Proyecto Fin de Carrera se ha desarrollado un módulo (o *problem type* en terminología de *GiD*®) que permite crear de forma sencilla el fichero de entrada para *NetGen* con la información de geometría de un proyecto concreto. Sería muy difícil que el usuario construyese este fichero de forma manual en proyectos muy complejos, por ejemplo si tuviera muchas entidades geométricas o se realizase algún tipo de operación entre ellas.

Como ya se ha comentado en capítulos anteriores se ha elegido *GiD*® como herramienta base por las características tan buenas que tiene su interfaz gráfica, además permite personalizar las opciones de sus menús para cubrir por completo las necesidades del usuario. Además, gracias a que el usuario dispondrá de un mallado generado con la herramienta *NetGen*, nos hemos permitido mostrar una comparativa entre este mallado y el que realmente proporcionaría *GiD*® bajo las mismas condiciones de configuración. Esta comparación se puede llevar a cabo gracias a que *GiD*® posee una herramienta que se encarga de obtener una serie de parámetros de calidad de los mallados.

No se ha pretendido determinar si un generador de mallado es mejor que el otro o no, ya que debido al gran número de opciones de configuración que posee *GiD*®, se puede llegar a convertir en un generador de mallado mucho mejor que *NetGen*, lo que se ha intentado es que un usuario pueda saber qué generador de mallado utilizar en cada caso en función de los resultados obtenidos en dicha comparación.

Mediante el módulo *GiDtoNet* se ha creado un nuevo menú que se incluye en los proporcionados por *GiD*®. La forma de mostrar las diferentes opciones que existen para realizar un proyecto con este módulo *GiDtoNet* es mediante el uso de ventanas.

El módulo *GiDtoNet* se ha desarrollado mediante un código Tcl-Tk legible, limpio y claro. Además, está completamente documentado pensando en futuras actualizaciones que se pudieran llevar a cabo por otros desarrolladores. De la misma forma, se ha cuidado la estructura del mismo, de manera que puede ser seguido por cualquier persona que lea este documento.

Las pruebas realizadas sobre el módulo *GiDtoNet* han sido profundas y exhaustivas, seleccionando aquéllas que pudieran aportar mayor claridad a los resultados. Las comparativas que se han realizado entre ambos generadores de mallado siempre han sido en función de los parámetros que nos proporciona *GiD*®, ya que era la única forma de poder tener una base robusta sobre la que desarrollar las conclusiones.

Por último hay que destacar que la aportación realizada con el módulo *GiDtoNet* a los usuarios de los generadores de mallado bajo estudio permitirá que sean ellos los que puedan crear el fichero de entrada a *NetGen* con la geometría de un problema concreto. Además, a parte del módulo en sí, con la comparativa que se ha creado, se aporta una información extra para que dichos usuarios sepan qué mallado sería mejor utilizar en cada caso concreto.

Una vez desarrolladas las pruebas del capítulo 5, llegamos a las siguientes conclusiones:

- En todas las entidades geométricas soportadas por *NetGen*, excepto en el tronco de cono, la comparativa realizada entre las mallas creadas con cada uno de los generadores de mallado da como resultado que tiene mejor calidad la malla creada con *NetGen*, con la ayuda del módulo *GiDtoNet*, que la generada directamente con *GiD*®.
- En el caso del tronco de cono, al tener representaciones diferentes con una y otra herramienta es más dificil llegar a una conclusión. Lo que queda patente es que la adaptatividad de la malla con *NetGen* a la estructura es mayor que con *GiD*®.
- A medida que el proyecto crece en complejidad, los resultados obtenidos en cuanto a las calidades de los mallados no son tan evidentes. La malla creada mediante el módulo *GiDtoNet* se adapta mucho mejor a las estructuras que la proporcionada por *GiD*®. Esta adaptatividad del mallado hace que se se generen un mayor número de elementos, por lo que el tiempo de procesado aumentará al resolver la estructura. En este tipo de problemas se recomienda al usuario que sea él el que priorice, en función de sus necesidades, los parámetros en los que se basará para elegir entre un generador de mallado y otro.
- Se ha comprobado también que al modificar los parámetros de configuración de los generadores de mallado se obtienen soluciones distintas. De esta forma, si el valor de grading es mayor que el tamaño de mallado, se obtienen peores resultados con *NetGen* que con *GiD*®.

Para terminar, se recuerda que este Proyecto Fin de Carrera aporta una herramienta muy útil para todos aquellos usuarios del generador de mallado *NetGen* que antes tenían que construir sus ficheros de geometría manualmente, ya que les ha facilitado su creación. Además, el estudio realizado entre ambos generadores de mallado permite que el usuario tenga una visión completa de las ventajas e inconvenientes de usar uno y otro a la hora de desarrollar sus propios proyectos.

6.2. Futuras Líneas de Investigación.

El módulo se encuentra en una versión estable que cuenta con todas las características ya mencionadas en este documento y que eran objetivo de este Proyecto Fin de Carrera. A pesar de no tener previsto continuar con la investigación siguiendo la línea que ha permitido la realización de este Proyecto, se proponen una serie de ampliaciones.

- Implementación en *GiDtoNet* de nuevas entidades geométricas simples no contempladas en este Proyecto Fin de Carrera.

- Implementación de operaciones entre más de dos entidades geométricas simples.

- Optimización de algunos de los algoritmos estes en el módulo. En concreto, optimizar la manera de actualizar la información de los elementos estes en el proyecto cuando se sufre algún cambio.

- Integración con los módulos que actualmente utilizan los usuarios de *GiD*®.

Capítulo 7

Presupuesto para la realización del proyecto.

En este capítulo se presenta el presupuesto asociado al desarrollo de este Proyecto Fin de Carrera. En primer lugar se detallan los costes de personal en que se ha incurrido y, a continuación, se muestran los costes derivados del material empleado. La adición de cada uno de estos costes constituirá el coste total.

7.1. Costes de personal.

Los costes de personal incluyen los honorarios del Ingeniero de Telecomunicación encargado del desarrollo del proyecto. Aunque el ritmo de trabajo no ha sido constante, se estima que se han invertido aproximadamente 5 horas al día durante los meses de diciembre de 2009 a septiembre de 2010. Esto hace un total de 1000 horas. Según la Universidad, los honorarios de un Ingeniero de Telecomunicación son 26.95 euros por hora.

Concepto	Horas	Honorarios	Importe
Ingeniero de Telecomunicación	1000	26.95 euros/hora	26950 euros

7.2. Costes materiales.

Los materiales empleados durante la realización del proyecto han sido los siguientes:

- Un ordenador portátil de sobremesa con sistema operativo Windows, valorado aproximadamente en 690 euros.

- El *software* utilizado está compuesto por dos herramientas principales, *GiD*® y *NetGen*. La primera de ellas necesita licencia, aunque para el desarrollo del proyecto se han utilizado unas licencias gratuítas de un mes que proporciona *GiD*®. *NetGen* es gratuíto.

- Conexión a Internet durante la realización del proyecto. Ha sido necesaria para conseguir documentación.

Concepto	Unidades	Precio unitario	Importe
Ordenador personal	1	690 euros	690 euros
Conexión a internet	10	39 euros/mes	390 euros

7.3. Presupuesto total.

El presupuesto final para la realización de este proyecto está formado por los costes de material y de personal presentados anteriormente. El total asciende a 28030 euros y se resumen en la siguiente tabla.

Concepto	Importe
Costes de personal	26950 euros
Costes materiales	1080 euros
TOTAL	28030 euros

Apéndice A

Procedimientos Tcl/Tk de *GiDtoNet*.

Procedimiento TCL	Parámetros de entrada	Definición del procedimiento
Add_cone	*id_volumen*. Entero que indica el número de volúmenes que se tienen que añadir de un tipo determinado.	Almacena toda la informacion referente a entidades geometricas que sean conos.
Add_cylinder	*id_volumen*. Entero que indica el número de volúmenes que se tienen que añadir de un tipo determinado.	Almacena toda la informacion referente a entidades geometricas que sean cilindros.
Add_prism	*id_volumen*. Entero que indica el número de volúmenes que se tienen que añadir de un tipo determinado.	Almacena toda la informacion referente a entidades geometricas que sean hexaedros de base cuadrada.

Add_sphere	*id_volumen*. Entero que indica el número de volúmenes que se tienen que añadir de un tipo determinado.	Almacena toda la informacion referente a entidades geometricas que sean esferas.
Add_volume	*volumes_number*. Entero que indica el número de volúmenes que se tienen que añadir de un tipo determinado.	Se ejecuta cuando se quiere añadir un nuevo volumen al proyecto, en este caso este procedimiento añade tantos volúmenes como el valor de su parámetro de entrada.
All_volumes	Ninguno	Recorre todo el espacion de entidades geométricas estes en el proyecto y vuelve a almacenar toda su información.
Assign_bc	*bc*. Entero que representa la condicion de contorno que el usuario quiere asignar a una superficie determinada.	Se encarga de asignar la condicion de contorno sobre la superficie que el usuario ha elegido.
Button_scape	Ninguno	se ejecuta al pulsar la tecla scape y que sirve para actualizar la información del proyecto cuando se ha producido un cambio en éste.

Cancel	Ninguno	Se ejecuta cuando el usuario quiere salir de la ventana de creación de volúmenes.
Close_window	*window*. Parámetro que indica la ventana.	Se ejecuta cuando el usuario quiere cerrar la ventana de creación de volúmenes.
Create_file	Ninguno	Escribe el fichero *CSG* con la info almacenada de cada uno de los volumenes del proyecto.
Create_GiD®mesh	Ninguno	Transforma el fichero *GiDtoNetNet.msh* que devuelve *NetGen* en un fichero con el mallado completo para que *GiD®* lo pueda importar.
Create_meshFile	Ninguno	Llama de forma remota al generador de mallado *NetGen* para obtener el fichero *.msh generado por este.
Create_volume	Ninguno	Carga la ventana con las opciones para crear volumnes y operaciones.
Escribir_script	Ninguno	Escribe el script para realizar la llamada remota al generador de mallado *NetGen*.

GetLineValues	Ninguno	Devuelve la información de una línea de un fichero en una lista.
Inicializar	*file*. Fichero del que queremos leer las líneas.	Extrae toda la información de volúmenes y nodos del fichero que se le pasa por parámetro, en nuestro caso el *GiDtoNetNet*.
InitGIDProject	*dir*. Dirección del módulo que se esta ejecutando.	Se ejecutar al cargar el módulo *GiDtoNet* y que crea la interfaz.
LoadGIDProject	*filespd*. Ruta en la que está almacenado el proyecto.	Se ejecuta cuando el usuario carga el proyecto, restaura la información que se almacenó de éste.
Operation	*op*. Entero que indica la operacion que el usuario ha seleccionado.	Realiza la operacion entre las entidades geometricas elegidas por el usuario.
Pulsa_boton	*tipo_volumen*. Entero que representa qué volumen se va a crear.	Se ejecuta cuando el usuario selecciona un volumen desde la ventana y lo crea dentro del proyecto de *GiD*®.
SaveGIDProject	*filespd*. Ruta en la que se va a almacenar el proyecto.	Se ejecuta cuando el usuario pulsa la opción de guardar el proyecto, almacena la información de éste.

Read_save_file	Ninguno	Lee y extrae del fichero GiDtoNetNet.txt la info de un proyecto que estaba guardado. Solamente se ejecuta al cargar un proyecto.
Window_bc	Ninguno	Crea la ventana para asignar las condiciones de contorno a las superficies de las entidades geométricas simples.
Write_save_file	Ninguno	Crea el fichero para almacenar la informacion este en el proyecto. Solamente se ejecuta cuando el usuario almacena el proyecto.

Apéndice B

Ficheros de *GiDtoNet.*

Nombre del fichero	Descripción
GiDtoNetNet.geo	Almacenas las primitivas de las distintas entidades geométricas implicadas en el proyecto.
GiDtoNetNet.msh	Contiene el mallado que ha creado *NetGen* del proyecto en el formato indicado por esta herramienta.
GiDtoNet.tcl	Contiene los procedimientos y variables Tcl que dan forma al módulo en el que se basa este Proyecto Fin de Carrera.
GiD2Net.msh	Contiene el mallado que hemos volcado desde el fichero *GiDtoNetNet.msh* para poder importarlo a *GiD*® pero con el mallado creado por *NetGen*.
Project_info.txt	Contiene la información que *GiD*® no es capaz de almacenar para poder cargar nuevamente el proyecto en el que estamos trabajando. Este fichero se refrescará cada vez que el usuario pulse la opción *Save*.
script.bat	Contiene una sentencia que utilizaremos para llamar de forma remota a *NetGen*.

Apéndice C

Código *GiDtoNet.*

```
#########################################################################
#                                                                       #
# GiDtoNet.tcl, 15/10/2010 Laura Vozmediano Latorre                     #
#                                                                       #
# Archivo TCL que contiene todos los procedimientos                     #
# que controlan el modulo GiDtoNet.                                     #
#                                                                       #
#########################################################################

#Estas variables estaran accesible durante todo el programa
namespace eval GID2NET {
  #almacena la informacion de los volumenes
  variable LIST_INFO
  #almacena la informacion para construir el fichero geo
  variable LIST_FILE
  #contador para el numero de volumenes del proyecto
  variable VOLUMES
  #identificador del ultimo volumen creado
  variable LAST_ID
  #numero de operaciones booleanas del proyect
  variable NUM_OP
  #lista con la informacion de las operaciones
  variable LIST_OPERATIONS
  #flag para saber si la ultima operacion fue una AND NOT
  variable AND_NOT
  #ruta en la que se ha creado el fichero geo
  variable RUTA_GEO
}

#Procedimiento encargado de cargar el modulo
proc InitGIDProject { dir } {

  global DIR
  set DIR $dir
  Inicializar
  #construccion del menu principal con las opciones del modulo
  GiD$^{\begin{small}\textregistered\end{small}}$Menu::Create "GID2NET" "PRE"

  GiD$^{\begin{small}\textregistered\end{small}}$Menu::InsertOption \
  "GID2NET" [list "Create volume"] \
    0 PRE "Create_volume"

  GiD$^{\begin{small}\textregistered\end{small}}$Menu::InsertOption \
   "GID2NET" [list "Assign boundary conditions"] \
    1 PRE "Window_bc"

  GiD$^{\begin{small}\textregistered\end{small}}$Menu::InsertOption \
```

```
  "GID2NET" [list "Create_file_.geo"] \
    2 PRE "Create_file"

  GiD$^{\begin{small}\textregistered\end{small}}$Menu::InsertOption \
  "GID2NET" [list "Create_mesh_file"] \
    3 PRE "Create_meshFile"

  GiD$^{\begin{small}\textregistered\end{small}}$Menu::UpdateMenus

  #relacionar un evento de teclado con un procedimiento
  bind .gid <KeyPress-Escape> "Button_scape"
  bind .gid <KeyRelease-Escape> "Button_scape"
}

#Permite inicializar las variables del modulo
proc Inicializar {} {
  set ::GID2NET::VOLUMES [GiD_Info Geometry NumVolumes]
  set ::GID2NET::AND_NOT 0
  set ::GID2NET::new_bc 0
  set ::GID2NET::LAST_ID 0
  set ::GID2NET::NUM_OP 0
}

#Cuando se guarda el proyecto se almacena la informacion importante
proc SaveGIDProject {filespd} {
  Write_save_file
}

#Se carga la informacion del proyecto si este estaba creado
proc LoadGIDProject { filespd } {

  global DIR
  set name [file join $DIR GiDtoNet.spd]
  set coincidencia ".gid"

  if {$name == $filespd} {
  } else {
    set indice_coincidencia [string first $coincidencia $filespd]
    set indice_real [expr $indice_coincidencia + 3]
    set project_name [string range $filespd 0 $indice_real]
    Read_save_file $project_name
  }
  All_volumes
}

#Procedimiento que se encargar de comprobar si hay cambios en el proyecto
proc Button_scape {} {
  global actual_type
  #valor de los volumenes actuales
  set actual_volumes [GiD_Info Geometry NumVolumes]
  #si se ha eliminado alguno se vuelven a crear
  if {$actual_volumes < $::GID2NET::VOLUMES} {
    All_volumes
  }
  if {$::GID2NET::AND_NOT == 1} {
    #la operacion and not se trata de forma diferente
    All_volumes
    set ::GID2NET::AND_NOT 0
  }
  if {$::GID2NET::new_bc == 1} {
    #se vuelven a obtener los valores de todos los volumenes porque ha habido cambios
    All_volumes
    set ::GID2NET::new_bc 0
  }
  #si se ha creado alguno hay que hacer su proceso de creacion
  if {$actual_volumes > $::GID2NET::VOLUMES} {
    #numero de nuevos volumenes
```

```
    set num_new_vol [expr $actual_volumes - $::GID2NET::VOLUMES]

    #actualizar la variable global
    set ::GID2NET::VOLUMES [expr $::GID2NET::VOLUMES + $num_new_vol]
    #crear los volumnes
    Add_volume $num_new_vol
  }
}

#Construye la ventana con las opciones necesarias para crear entidades
proc Create_volume {} {

  global DIR

  #Cargar las imagenes de los botones
  set image_button_prism [image create photo -file [file join $DIR Images hexaedro.gif]]
  set image_button_esfera [image create photo -file [file join $DIR Images esfera.gif]]
  set image_button_cilin [image create photo -file [file join $DIR Images cilindro.gif]]
  set image_button_cono [image create photo -file [file join $DIR Images cono.gif]]

  #Crear la ventana
  set w .gid.win_example
  InitWindow $w "Create Volume" c_volum "" "" 1

  #Crear los marcos que dividen la ventana en opciones
  labelframe $w.volumes -text [= "Volumes to create entities"] -borderwidth 3 -padx 20 -pady 10
  labelframe $w.operations -text [= "Operations availables"] -borderwidth 3 -padx 20 -pady 10
  labelframe $w.exit -borderwidth 3 -padx 20 -pady 10

  #Texto que se imprimira en la ventana
  label $w.volumes.etiqueta -text "SELECT THE VOLUME YOU WANT \n"

  #Creacion de botones
  button $w.volumes.prism -image $image_button_prism -height 80 \
-width 50 -command "Pulsa boton Prism"
  button $w.volumes.sphere -image $image_button_esfera -height 80 \
-width 80 -command "Pulsa boton Sphere"
  button $w.volumes.cilinder -image $image_button_cilin -height 80 \
-width 50 -command "Pulsa boton Cylinder"
  button $w.volumes.cone -image $image_button_cono -height 80 \
-width 50 -command "Pulsa boton Cone"
  button $w.exit.cancel -text [= "CANCEL"] -height 2 -width 10 -command "Cancel $w"
  button $w.exit.close -text [= "CLOSE"] -height 2 -width 10 -command "Close_window $w"
  button $w.operations.and -text [= "AND"] -height 2 -width 10 -command "Operation 1"
  button $w.operations.or -text [= "OR"] -height 2 -width 10 -command "Operation 2"
  button $w.operations.not -text [= "NOT"] -height 2 -width 10 -command "Operation 3"

  #Mostrar todos los elementos en la ventana
  pack $w.volumes.etiqueta -anchor center -pady 5
  pack $w.volumes.prism $w.volumes.sphere $w.volumes.cilinder $w.volumes.cone \
-side left -anchor center
  pack $w.operations.and $w.operations.or $w.operations.not -side left -anchor center
  pack $w.exit.cancel $w.exit.close -side right -anchor center
  pack $w.volumes
  pack $w.operations -pady 20
  pack $w.exit -pady 10
}

#Crea la ventana para asignar las condiciones de contorno
proc Window_bc {} {

  #Cargar valores de variables globales
  global DIR
  global OTRA_BC

  #Cargar las imagenes de los botones
  set cero [image create photo -file [file join $DIR Images cero.tif]]
```

```
  set uno [image create photo -file [file join $DIR Images uno.tif]]
  set dos [image create photo -file [file join $DIR Images dos.tif]]
  set tres [image create photo -file [file join $DIR Images tres.tif]]
  set asignar [image create photo -file [file join $DIR Images assign.tif]]
  set colores [image create photo -file [file join $DIR Images colors.tif]]
  set num [image create photo -file [file join $DIR Images numbers.tif]]
  set cerrar [image create photo -file [file join $DIR Images close.tif]]

  #Construir la ventana
  set w .gid.boundary_conditions
  InitWindow $w "Assign Boundary Condition" c_volum "" "" 20

  #Marcos para separar las opciones de la ventana
  labelframe $w.choices -text [= "Initial Conditions"]
  labelframe $w.free -text [= "Other Conditions"] -borderwidth 3 -padx 20 -pady 10
  labelframe $w.draw -text [= "Draw conditions by"] -borderwidth 3 -padx 20 -pady 10
  labelframe $w.exit -borderwidth 3 -padx 5 -pady 5

  #Construccion de los botones
  button $w.choices.zero -image $cero -height 35 -width 60 -command "Assign_bc 0"
  button $w.choices.one -image $uno -height 35 -width 60 -command "Assign_bc 1"
  button $w.choices.two -image $dos -height 35 -width 60 -command "Assign_bc 2"
  button $w.choices.three -image $tres -height 35 -width 60 -command "Assign_bc 3"
  button $w.draw.color -image $colores -height 40 -width 85 -command "View_colors_bc"
  button $w.draw.number -image $num -height 40 -width 90 -command "View_numbers_bc"
  button $w.exit.close -image $cerrar -height 35 -width 60 -command "destroy $w"
  button $w.free.assign -image $asignar -height 35 -width 85 -command "Assign_bc 4"

  #Texto que se imprimira en la ventana
  label $w.free.etiqueta -text [= "For another bc's flag:"] \
  -justify center

  set OTRA_BC ""

  #Creacion de entry para nuevas condiciones de contorno
  entry $w.free.otra -bg white -relief sunken\
      -textvariable OTRA_BC -width 6 -justify center

  #Mostrar todos los elementos en la ventana
  pack $w.choices.zero -side left -anchor w -padx 10 -pady 10
  pack $w.choices.one -side left -anchor w -padx 10 -pady 10
  pack $w.choices.two -side left -anchor w -padx 10 -pady 10
  pack $w.choices.three -side left -anchor w -padx 10 -pady 10
  pack $w.draw.color -side left -anchor w -padx 20
  pack $w.draw.number -side left -anchor w
  pack $w.exit.close -side bottom -anchor w
  pack $w.free.etiqueta -side left
  pack $w.free.otra -side left -padx 20 -pady 10
  pack $w.free.assign -side right
  pack $w.choices -side top
  pack $w.free -side top
  pack $w.exit -side bottom -pady 10
  pack $w.draw -side bottom
}

#Metodo que se ejecuta al pulsar un boton para crear un volumen
proc Pulsa_boton {tipo_volumen} {
  #almacena el tipo de volumen actual
  global actual_type
  #fuerza que se guarde toda la info antes de crear el volumen
  Button_scape
  set actual_type $tipo_volumen
  #se crea el objeto
  GiD_Process Mescape Geometry Create Object $tipo_volumen
}
```

```
#Almacena la informacion que haya o interrumpe una accion
proc Cancel {window} {
  Button_scape
}

#Cierra la ventana y almacena toda la informacion del proyecto
proc Close_window {window} {
  Button_scape
  destroy $window
}

#Metodo que permite almacenar toda la informacion tras un cambio
proc All_volumes {} {
  #se 'borran' todas las operaciones
  for {set i 0} {$i < $::GID2NET::NUM_OP} {incr i} {
    set ::GID2NET::LIST_OPERATIONS(BORRADO,$i) 1
  }
  #numero de volumenes actual
  set actual_volumes [GiD_Info Geometry NumVolumes]
  #comienza de nuevo la cuenta de volumenes
  set ::GID2NET::VOLUMES 0
  #lista con las condiciones de GID de los volumenes presentes
  set conditions [list]
  #Extraemos una lista con las condiciones de todos los volumenes presentes
  for {set i 0} {$i < $::GID2NET::LAST_ID} {incr i} {
    set id_volumen [expr $i + 1]
    lappend conditions [GiD_Info Conditions Type geometry $id_volumen]
  }
  #recorremos todas las condiciones y buscamos que tipo de volumen es
  #se vuelven a rellenar todos los arrays
  for {set i 0} {$i < $::GID2NET::LAST_ID} {incr i} {
    set cond_actual [lindex $conditions $i]
    #se obtiene el identificador del volumen para saber si se ha borrado o no
    set id_volumen [expr $i + 1]
    if {$cond_actual == "{E $id_volumen - 0}"} {
      Add_prism $id_volumen
      set ::GID2NET::VOLUMES [expr $::GID2NET::VOLUMES + 1]
    } elseif {$cond_actual == "{E $id_volumen - 1}"} {
      Add_sphere $id_volumen
      set ::GID2NET::VOLUMES [expr $::GID2NET::VOLUMES + 1]
    } elseif {$cond_actual == "{E $id_volumen - 2}"} {
      Add_cylinder $id_volumen
      set ::GID2NET::VOLUMES [expr $::GID2NET::VOLUMES + 1]
    } elseif {$cond_actual == "{E $id_volumen - 3}"} {
      Add_cone $id_volumen
      set ::GID2NET::VOLUMES [expr $::GID2NET::VOLUMES + 1]
    #si es un volumen que procede de una operacion se vuelve a rellenar la lista de la operacion
    } elseif {$cond_actual == "{E $id_volumen - 4}"} {
      #Tenemos que volver a almacenar las operaciones por si se borra alguna
      for {set j 0} {$j < $::GID2NET::NUM_OP} {incr j} {
        if {$::GID2NET::LIST_OPERATIONS(NUM_VOL,$j) == $id_volumen} {
          set ::GID2NET::LIST_OPERATIONS(BORRADO,$j) 0
          set ::GID2NET::VOLUMES [expr $::GID2NET::VOLUMES + 1]
        }
      }
    } elseif {$cond_actual == "{E $id_volumen - 5}"} {
      for {set j 0} {$j < $::GID2NET::NUM_OP} {incr j} {
        if {$::GID2NET::LIST_OPERATIONS(NUM_VOL,$j) == $id_volumen} {
          set ::GID2NET::LIST_OPERATIONS(BORRADO,$j) 0
          set ::GID2NET::VOLUMES [expr $::GID2NET::VOLUMES + 1]
        }
      }
    } elseif {$cond_actual == "{E $id_volumen - 6}"} {
      set ::GID2NET::LIST_INFO(TYPE,$i) ""
      set ::GID2NET::VOLUMES [expr $::GID2NET::VOLUMES + 1]

    } else {
```

```
      set ::GID2NET::LIST_INFO(TYPE,$i) ""
    }

  }
}

#Annade un volumen o mas y les asigna un identificador
proc Add_volume {volumes_number} {
  global actual_type
  for {set i 0} {$i < $volumes_number} {incr i} {
    if {$actual_type == "Prism"} {
      set id [expr $::GID2NET::LAST_ID + 1]
      Add_prism $id
      set ::GID2NET::LAST_ID [expr $::GID2NET::LAST_ID + 1]
    }
    if {$actual_type == "Sphere"} {
      set id [expr $::GID2NET::LAST_ID + 1]
      Add_sphere $id
      set ::GID2NET::LAST_ID [expr $::GID2NET::LAST_ID + 1]
    }
    if {$actual_type == "Cylinder"} {
      set id [expr $::GID2NET::LAST_ID + 1]
      Add_cylinder $id
      set ::GID2NET::LAST_ID [expr $::GID2NET::LAST_ID + 1]
    }
    if {$actual_type == "Cone"} {
      set id [expr $::GID2NET::LAST_ID + 1]
      Add_cone $id
      set ::GID2NET::LAST_ID [expr $::GID2NET::LAST_ID + 1]
    }
  }
}

#Annade un hexaedro
proc Add_prism {id_volumen} {
  #indice del array
  set indice [expr $id_volumen - 1]
  #asignar la condicion GID al volumen
  GiD_AssignData condition Type Volumes 0 $id_volumen
  #extraer la info del volumen en un lista
  set information_volumen [GiD_Info list_entities Volumes $id_volumen -sublist]
  set s1 [lindex $information_volumen 14]
  set s2 [lindex $information_volumen 18]
  set s3 [lindex $information_volumen 22]
  set s4 [lindex $information_volumen 26]
  set s5 [lindex $information_volumen 30]
  set s6 [lindex $information_volumen 34]
  #informacion de caras
  set inf_s1 [GiD_Info list_entities surfaces $s1 -sublist]
  set inf_s2 [GiD_Info list_entities surfaces $s2 -sublist]
  set inf_s3 [GiD_Info list_entities surfaces $s3 -sublist]
  set inf_s4 [GiD_Info list_entities surfaces $s4 -sublist]
  set inf_s5 [GiD_Info list_entities surfaces $s5 -sublist]
  set inf_s6 [GiD_Info list_entities surfaces $s6 -sublist]
  #info de las condiciones de contorno
  for {set i 0} {$i < 6} {incr i} {
    if {$i == 0} {
      set superf $s1
    }
    if {$i == 1} {
      set superf $s2
    }
    if {$i == 2} {
      set superf $s3
    }
    if {$i == 3} {
      set superf $s4
```

```
    }
    if {$i == 4} {
      set superf $s5
    }
    if {$i == 5} {
      set superf $s6
    }
    set dato_bc [GiD_Info Conditions Boundary geometry $superf]
    set long_string [string length $dato_bc]
    set valor_real [string index $dato_bc [expr $long_string - 2]]
    if {$valor_real == "-"} {
      set valor_real 0
    }
    if {$i == 0} {
      set bc_s1 $valor_real
    }
    if {$i == 1} {
      set bc_s2 $valor_real
    }
    if {$i == 2} {
      set bc_s3 $valor_real
    }
    if {$i == 3} {
      set bc_s4 $valor_real
    }
    if {$i == 4} {
      set bc_s5 $valor_real
    }
    if {$i == 5} {
      set bc_s6 $valor_real
    }
}
#informacion de los puntos
for {set i 0} {$i < 3} {incr i} {
  set center_s1($i) [lindex $inf_s1 [expr 91 + $i]]
}
for {set i 0} {$i < 3} {incr i} {
  set center_s2($i) [lindex $inf_s2 [expr 91 + $i]]
}
for {set i 0} {$i < 3} {incr i} {
  set center_s3($i) [lindex $inf_s3 [expr 91 + $i]]
}
for {set i 0} {$i < 3} {incr i} {
  set center_s4($i) [lindex $inf_s4 [expr 91 + $i]]
}
for {set i 0} {$i < 3} {incr i} {
  set center_s5($i) [lindex $inf_s5 [expr 91 + $i]]
}
for {set i 0} {$i < 3} {incr i} {
  set center_s6($i) [lindex $inf_s6 [expr 91 + $i]]
}
for {set i 0} {$i < 3} {incr i} {
  set normal_s1($i) [expr [lindex $inf_s1 [expr 95 + $i]]*(-1)]
}
for {set i 0} {$i < 3} {incr i} {
  set normal_s2($i) [expr [lindex $inf_s2 [expr 95  + $i]]*(-1)]
}
for {set i 0} {$i < 3} {incr i} {
  set normal_s3($i) [expr [lindex $inf_s3 [expr 95 + $i]]*(-1)]
}
for {set i 0} {$i < 3} {incr i} {
  set normal_s4($i) [expr [lindex $inf_s4 [expr 95 + $i]]*(-1)]
}
for {set i 0} {$i < 3} {incr i} {
  set normal_s5($i) [expr [lindex $inf_s5 [expr 95 + $i]]*(-1)]
}
for {set i 0} {$i < 3} {incr i} {
```

```
    set normal_s6($i) [expr [lindex $inf_s6 [expr 95 + $i]]*(-1)]
  }
  #completar las listas con la info que se almacena de los volumenes
  set ::GID2NET::LIST_INFO(TYPE,$indice) 0
    set ::GID2NET::LIST_INFO(CENTER_S1_X,$indice) $center_s1(0)
  set ::GID2NET::LIST_INFO(CENTER_S1_Y,$indice) $center_s1(1)
  set ::GID2NET::LIST_INFO(CENTER_S1_Z,$indice) $center_s1(2)
    set ::GID2NET::LIST_INFO(CENTER_S2_X,$indice) $center_s2(0)
  set ::GID2NET::LIST_INFO(CENTER_S2_Y,$indice) $center_s2(1)
  set ::GID2NET::LIST_INFO(CENTER_S2_Z,$indice) $center_s2(2)
    set ::GID2NET::LIST_INFO(CENTER_S3_X,$indice) $center_s3(0)
  set ::GID2NET::LIST_INFO(CENTER_S3_Y,$indice) $center_s3(1)
  set ::GID2NET::LIST_INFO(CENTER_S3_Z,$indice) $center_s3(2)
    set ::GID2NET::LIST_INFO(CENTER_S4_X,$indice) $center_s4(0)
  set ::GID2NET::LIST_INFO(CENTER_S4_Y,$indice) $center_s4(1)
  set ::GID2NET::LIST_INFO(CENTER_S4_Z,$indice) $center_s4(2)
    set ::GID2NET::LIST_INFO(CENTER_S5_X,$indice) $center_s5(0)
  set ::GID2NET::LIST_INFO(CENTER_S5_Y,$indice) $center_s5(1)
  set ::GID2NET::LIST_INFO(CENTER_S5_Z,$indice) $center_s5(2)
    set ::GID2NET::LIST_INFO(CENTER_S6_X,$indice) $center_s6(0)
  set ::GID2NET::LIST_INFO(CENTER_S6_Y,$indice) $center_s6(1)
  set ::GID2NET::LIST_INFO(CENTER_S6_Z,$indice) $center_s6(2)
  set ::GID2NET::LIST_INFO(NORMAL_S1_X,$indice) $normal_s1(0)
  set ::GID2NET::LIST_INFO(NORMAL_S1_Y,$indice) $normal_s1(1)
  set ::GID2NET::LIST_INFO(NORMAL_S1_Z,$indice) $normal_s1(2)
  set ::GID2NET::LIST_INFO(NORMAL_S2_X,$indice) $normal_s2(0)
  set ::GID2NET::LIST_INFO(NORMAL_S2_Y,$indice) $normal_s2(1)
  set ::GID2NET::LIST_INFO(NORMAL_S2_Z,$indice) $normal_s2(2)
  set ::GID2NET::LIST_INFO(NORMAL_S3_X,$indice) $normal_s3(0)
  set ::GID2NET::LIST_INFO(NORMAL_S3_Y,$indice) $normal_s3(1)
  set ::GID2NET::LIST_INFO(NORMAL_S3_Z,$indice) $normal_s3(2)
  set ::GID2NET::LIST_INFO(NORMAL_S4_X,$indice) $normal_s4(0)
  set ::GID2NET::LIST_INFO(NORMAL_S4_Y,$indice) $normal_s4(1)
  set ::GID2NET::LIST_INFO(NORMAL_S4_Z,$indice) $normal_s4(2)
  set ::GID2NET::LIST_INFO(NORMAL_S5_X,$indice) $normal_s5(0)
  set ::GID2NET::LIST_INFO(NORMAL_S5_Y,$indice) $normal_s5(1)
  set ::GID2NET::LIST_INFO(NORMAL_S5_Z,$indice) $normal_s5(2)
  set ::GID2NET::LIST_INFO(NORMAL_S6_X,$indice) $normal_s6(0)
  set ::GID2NET::LIST_INFO(NORMAL_S6_Y,$indice) $normal_s6(1)
  set ::GID2NET::LIST_INFO(NORMAL_S6_Z,$indice) $normal_s6(2)
  #almaceno las BC
  set ::GID2NET::LIST_INFO(BC_S1,$indice) $bc_s1
  set ::GID2NET::LIST_INFO(BC_S2,$indice) $bc_s2
  set ::GID2NET::LIST_INFO(BC_S3,$indice) $bc_s3
  set ::GID2NET::LIST_INFO(BC_S4,$indice) $bc_s4
  set ::GID2NET::LIST_INFO(BC_S5,$indice) $bc_s5
  set ::GID2NET::LIST_INFO(BC_S6,$indice) $bc_s6
  #Almacenamos los planos que forman el hexaedro
  set ::GID2NET::LIST_INFO(PLANE1,$indice) "plane($center_s1(0),$center_s1(1),$center_s1(2);\
$normal_s1(0),$normal_s1(1),$normal_s1(2)) -bc = $bc_s1"
  set ::GID2NET::LIST_INFO(PLANE2,$indice) "plane($center_s2(0),$center_s2(1),$center_s2(2);\
$normal_s2(0),$normal_s2(1),$normal_s2(2)) -bc = $bc_s2"
  set ::GID2NET::LIST_INFO(PLANE3,$indice) "plane($center_s3(0),$center_s3(1),$center_s3(2);\
$normal_s3(0),$normal_s3(1),$normal_s3(2)) -bc = $bc_s3"
  set ::GID2NET::LIST_INFO(PLANE4,$indice) "plane($center_s4(0),$center_s4(1),$center_s4(2);\
$normal_s4(0),$normal_s4(1),$normal_s4(2)) -bc = $bc_s4"
  set ::GID2NET::LIST_INFO(PLANE5,$indice) "plane($center_s5(0),$center_s5(1),$center_s5(2);\
$normal_s5(0),$normal_s5(1),$normal_s5(2)) -bc = $bc_s5"
  set ::GID2NET::LIST_INFO(PLANE6,$indice) "plane($center_s6(0),$center_s6(1),$center_s6(2);\
$normal_s6(0),$normal_s6(1),$normal_s6(2)) -bc = $bc_s6"
  set ::GID2NET::LIST_FILE($indice) "p1_$indice and p2_$indice and p3_$indice and \
p4_$indice and p5_$indice and p6_$indice"
}

#Annade una esfera
proc Add_sphere {id_volumen} {
```

```
#indice del array
set indice [expr $id_volumen - 1]
#asignar la condicion GID al volumen
GiD_AssignData condition Type Volumes 1 $id_volumen
#extraer la info del volumen
set information_volumen [GiD_Info list_entities Volumes $id_volumen -sublist]
set s1 [lindex $information_volumen 14]
set s2 [lindex $information_volumen 18]
set s3 [lindex $information_volumen 22]
set s4 [lindex $information_volumen 26]
#extraer informacion importante
for {set i 0} {$i < 3} {incr i} {
  set center($i) [lindex $information_volumen [expr 30 + $i]]
}
set inf_s1 [GiD_Info list_entities surfaces $s1 -sublist]
set line1 [lindex $inf_s1 14]
set inf_l1 [GiD_Info list_entities lines $line1 -sublist]
set info_point1 [GiD_Info list_entities points [lindex $inf_l1 12] -sublist]
set info_point2 [GiD_Info list_entities points [lindex $inf_l1 13] -sublist]
for {set i 0} {$i < 3} {incr i} {
  set point1($i) [lindex $info_point1 [expr 12 + $i]]
}
for {set i 0} {$i < 3} {incr i} {
  set point2($i) [lindex $info_point2 [expr 12 + $i]]
}
for {set i 0} {$i < 3} {incr i} {
  set resta($i) [expr $point1($i) - $center($i)]
}
for {set i 0} {$i < 4} {incr i} {
  if {$i == 0} {
    set superf $s1
  }
  if {$i == 1} {
    set superf $s2
  }
  if {$i == 2} {
    set superf $s3
  }
  if {$i == 3} {
    set superf $s4
  }
  #extraer condiciones de contorno
  set dato_bc [GiD_Info Conditions Boundary geometry $superf]
  set long_string [string length $dato_bc]
  set valor_real [string index $dato_bc [expr $long_string - 2]]
  if {$valor_real == "-"} {
    set valor_real 0
  }
  if {$i == 0} {
    set bc_s1 $valor_real
  }
  if {$i == 1} {
    set bc_s2 $valor_real
  }
  if {$i == 2} {
    set bc_s3 $valor_real
  }
  if {$i == 3} {
    set bc_s4 $valor_real
  }
}
if {$bc_s1 > 0} {
  set bc $bc_s1
} elseif {$bc_s2 > 0} {
  set bc $bc_s2
} elseif {$bc_s3 > 0} {
  set bc $bc_s3
```

```
  } elseif {$bc_s4 > 0} {
    set bc $bc_s4
  } else {
    set bc 0
  }
  GiD_AssignData condition Boundary Surfaces $valor_real $s1
  GiD_AssignData condition Boundary Surfaces $valor_real $s2
  GiD_AssignData condition Boundary Surfaces $valor_real $s3
  GiD_AssignData condition Boundary Surfaces $valor_real $s4
  #completar las listas con la info que se va a almacenar del volumen
  set radio [expr sqrt($resta(0)*$resta(0)+$resta(1)*$resta(1)+$resta(2)*$resta(2))]
  set ::GID2NET::LIST_INFO(TYPE,$indice) 1
  set ::GID2NET::LIST_INFO(CENTER_X,$indice) $center(0)
  set ::GID2NET::LIST_INFO(CENTER_Y,$indice) $center(1)
  set ::GID2NET::LIST_INFO(CENTER_Z,$indice) $center(2)
  set ::GID2NET::LIST_INFO(RADIO,$indice) $radio
  #almacenar bc
  set ::GID2NET::LIST_INFO(BC,$indice) $bc
  if {$bc == 0} {
    set ::GID2NET::LIST_FILE($indice) "sphere (($center(0),$center(1),$center(2);$radio)"
  } else {
    set ::GID2NET::LIST_FILE($indice) "sphere (($center(0),$center(1),$center(2);$radio) -bc = $bc"
  }
}

#Annadir un cilindro
proc Add_cylinder {id_volumen} {
  #indice del array
  set indice [expr $id_volumen - 1]
  #Condicion de GID par ael volumen
  GiD_AssignData condition Type Volumes 2 $id_volumen
  #extraer info del volumen
  set information_volumen [GiD_Info list_entities Volumes $id_volumen -sublist]
  set lateral1 [lindex $information_volumen 14]
  set lateral2 [lindex $information_volumen 18]
  set s1 [lindex $information_volumen 22]
  set s2 [lindex $information_volumen 26]
  set inf_s1 [GiD_Info list_entities surfaces $s1 -sublist]
  set inf_s2 [GiD_Info list_entities surfaces $s2 -sublist]
  #extraer informacion importante
  for {set i 0} {$i < 3} {incr i} {
    set center_s1($i) [lindex $inf_s1 [expr 83 + $i]]
  }
  for {set i 0} {$i < 3} {incr i} {
    set center_s2($i) [lindex $inf_s2 [expr 83 + $i]]
  }
  for {set i 0} {$i < 3} {incr i} {
    set normal_s1($i) [expr [lindex $inf_s1 [expr 87 + $i]]*(-1)]
  }
  for {set i 0} {$i < 3} {incr i} {
    set normal_s2($i) [expr [lindex $inf_s2 [expr 87 + $i]]*(-1)]
  }
  set line1 [lindex $inf_s1 14]
  set inf_l1 [GiD_Info list_entities lines $line1 -sublist]
  set info_point1 [GiD_Info list_entities points [lindex $inf_l1 12] -sublist]
  for {set i 0} {$i < 3} {incr i} {
    set point1($i) [lindex $info_point1 [expr 12 + $i]]
  }
  for {set i 0} {$i < 3} {incr i} {
    set resta($i) [expr $point1($i) - $center_s1($i)]
  }
  for {set i 0} {$i < 4} {incr i} {
    if {$i == 0} {
      set superf $lateral1
    }
    if {$i == 1} {
      set superf $lateral2
```

```
    }
    if {$i == 2} {
      set superf $s1
    }
    if {$i == 3} {
      set superf $s2
    }
    #extraer las condiciones de contorno
    set dato_bc [GiD_Info Conditions Boundary geometry $superf]
    set long_string [string length $dato_bc]
    set valor_real [string index $dato_bc [expr $long_string - 2]]
    if {$valor_real == "-"} {
      set valor_real 0
    }
    if {$i == 0} {
      set bc_lateral1 $valor_real
      GiD_AssignData condition Boundary Surfaces $valor_real $lateral1
      GiD_AssignData condition Boundary Surfaces $valor_real $lateral2
    }
    if {$i == 1} {
      set bc_lateral2 $valor_real
      GiD_AssignData condition Boundary Surfaces $valor_real $lateral1
      GiD_AssignData condition Boundary Surfaces $valor_real $lateral2
    }
    if {$i == 2} {
      set bc_s1 $valor_real
    }
    if {$i == 3} {
      set bc_s2 $valor_real
    }
  }
  if {$bc_lateral1 > 0} {
    set bc_lateral $bc_lateral1
  } elseif {$bc_lateral2 > 0} {
    set bc_lateral $bc_lateral2
  } else {
    set bc_lateral 0
  }

  #almacenar la info del volumen
  set radio [expr sqrt($resta(0)*$resta(0)+$resta(1)*$resta(1)+$resta(2)*$resta(2))]
  set ::GID2NET::LIST_INFO(TYPE,$indice) 2
    set ::GID2NET::LIST_INFO(C1_X,$indice) $center_s1(0)
  set ::GID2NET::LIST_INFO(C1_Y,$indice) $center_s1(1)
  set ::GID2NET::LIST_INFO(C1_Z,$indice) $center_s1(2)
  set ::GID2NET::LIST_INFO(C2_X,$indice) $center_s2(0)
  set ::GID2NET::LIST_INFO(C2_Y,$indice) $center_s2(1)
  set ::GID2NET::LIST_INFO(C2_Z,$indice) $center_s2(2)
  set ::GID2NET::LIST_INFO(RADIO,$indice) $radio
  set ::GID2NET::LIST_INFO(N_S1_X,$indice) $normal_s1(0)
  set ::GID2NET::LIST_INFO(N_S1_Y,$indice) $normal_s1(1)
  set ::GID2NET::LIST_INFO(N_S1_Z,$indice) $normal_s1(2)
  set ::GID2NET::LIST_INFO(N_S2_X,$indice) $normal_s2(0)
  set ::GID2NET::LIST_INFO(N_S2_Y,$indice) $normal_s2(1)
  set ::GID2NET::LIST_INFO(N_S2_Z,$indice) $normal_s2(2)
  #almacenar las condiciones de contorno
  set ::GID2NET::LIST_INFO(BC_S1,$indice) $bc_s1
  set ::GID2NET::LIST_INFO(BC_S2,$indice) $bc_s2
  set ::GID2NET::LIST_INFO(BC_LATERAL,$indice) $bc_lateral
  if {$bc_s1 == 0} {
    set ::GID2NET::LIST_INFO(PLANE1,$indice) "plane($center_s1(0),$center_s1(1),$center_s1(2);\
$normal_s1(0),$normal_s1(1),$normal_s1(2))"
  } else {
    set ::GID2NET::LIST_INFO(PLANE1,$indice) "plane($center_s1(0),$center_s1(1),$center_s1(2);\
$normal_s1(0),$normal_s1(1),$normal_s1(2)) -bc = $bc_s1"
  }
  if {$bc_s2 == 0} {
```

```
    set ::GID2NET::LIST_INFO(PLANE2,$indice) "plane($center_s2(0),$center_s2(1),$center_s2(2);\
$normal_s2(0),$normal_s2(1),$normal_s2(2))"
  } else {
    set ::GID2NET::LIST_INFO(PLANE2,$indice) "plane($center_s2(0),$center_s2(1),$center_s2(2);\
$normal_s2(0),$normal_s2(1),$normal_s2(2)) -bc = $bc_s2"
  }
  if {$bc_lateral == 0} {
    set ::GID2NET::LIST_FILE($indice) "cylinder($center_s1(0),$center_s1(1),$center_s1(2);\
$center_s2(0),$center_s2(1),$center_s2(2);$radio)"
  } else {
    set ::GID2NET::LIST_FILE($indice) "cylinder($center_s1(0),$center_s1(1),$center_s1(2);\
$center_s2(0),$center_s2(1),$center_s2(2);$radio) -bc = $bc_lateral"
  }
}

#Annadir un cono
proc Add_cone {id_volumen} {
  #indice del array
  set indice [expr $id_volumen - 1]
  #condicion de GID para el columen
  GiD_AssignData condition Type Volumes 3 $id_volumen
  #extraer la info del volumen
  set information_volumen [GiD_Info list_entities Volumes $id_volumen -sublist]
  set s1 [lindex $information_volumen 14]
  set lateral1 [lindex $information_volumen 18]
  set lateral2 [lindex $information_volumen 22]
  set s2 [lindex $information_volumen 26]
  set inf_s1 [GiD_Info list_entities surfaces $s1 -sublist]
  set inf_s2 [GiD_Info list_entities surfaces $s2 -sublist]
  #extraer la informacion importante
  for {set i 0} {$i < 3} {incr i} {
    set center_base($i) [lindex $inf_s2 [expr 87 + $i]]
  }
  for {set i 0} {$i < 3} {incr i} {
    set normal_base($i) [expr [lindex $inf_s2 [expr 91 + $i]]*(-1)]
  }
  set line1 [lindex $inf_s1 14]
  set line2 [lindex $inf_s2 14]
  set inf_l1 [GiD_Info list_entities lines $line1 -sublist]
  set inf_l2 [GiD_Info list_entities lines $line2 -sublist]
  set info_point1 [GiD_Info list_entities points [lindex $inf_l1 12] -sublist]
  set info_point2 [GiD_Info list_entities points [lindex $inf_l1 13] -sublist]
  for {set i 0} {$i < 3} {incr i} {
    set point1($i) [lindex $info_point1 [expr 12 + $i]]
  }
  for {set i 0} {$i < 3} {incr i} {
    set point2($i) [lindex $info_point2 [expr 12 + $i]]
  }
  set dist(0) [expr ($point2(0) - $center_base(0))/3]
  set dist(1) [expr ($point2(1) - $center_base(1))/3]
  set dist(2) [expr ($point2(2) - $center_base(2))/3]
  set point_plane(0) [expr $center_base(0) + (2*$dist(0))]
  set point_plane(1) [expr $center_base(1) + (2*$dist(1))]
  set point_plane(2) [expr $center_base(2) + (2*$dist(2))]
  for {set i 0} {$i < 3} {incr i} {
    set resta($i) [expr $point1($i) - $center_base($i)]
  }
  for {set i 0} {$i < 4} {incr i} {
    if {$i == 0} {
      set superf $s1
    }
    if {$i == 1} {
      set superf $lateral1
    }
    if {$i == 2} {
      set superf $lateral1
    }
```

```
    if {$i == 3} {
      set superf $s2
    }
    #extraer las condiciones de contorno
    set dato_bc [GiD_Info Conditions Boundary geometry $superf]
    set long_string [string length $dato_bc]
    set valor_real [string index $dato_bc [expr $long_string - 2]]
    if {$valor_real == "-"} {
      set valor_real 0
    }
    if {$i == 0} {
      set bc_s1 $valor_real
      GiD_AssignData condition Boundary Surfaces $valor_real $lateral1
      GiD_AssignData condition Boundary Surfaces $valor_real $lateral2
      GiD_AssignData condition Boundary Surfaces $valor_real $s1
    }
    if {$i == 1} {
      set bc_lateral1 $valor_real
      GiD_AssignData condition Boundary Surfaces $valor_real $lateral1
      GiD_AssignData condition Boundary Surfaces $valor_real $lateral2
      GiD_AssignData condition Boundary Surfaces $valor_real $s1
    }
    if {$i == 2} {
      set bc_lateral2 $valor_real
      GiD_AssignData condition Boundary Surfaces $valor_real $lateral1
      GiD_AssignData condition Boundary Surfaces $valor_real $lateral2
      GiD_AssignData condition Boundary Surfaces $valor_real $s1
    }
    if {$i == 3} {
      set bc_s2 $valor_real
    }
  }
  if {$bc_s1 > 0} {
    set bc_lateral $bc_s1
  } elseif {$bc_lateral1 > 0} {
    set bc_lateral $bc_lateral1
  } elseif {$bc_lateral2 > 0} {
    set bc_lateral $bc_lateral2
  } else {
    set bc_lateral 0
  }
  #almacenar la info del volumen en las listas
  set radio_base [expr sqrt($resta(0)*$resta(0)+$resta(1)*$resta(1)+$resta(2)*$resta(2))]
  set ::GID2NET::LIST_INFO(TYPE,$indice) 3
    set ::GID2NET::LIST_INFO(CENTER1_X,$indice) $center_base(0)
  set ::GID2NET::LIST_INFO(CENTER1_Y,$indice) $center_base(1)
  set ::GID2NET::LIST_INFO(CENTER1_Z,$indice) $center_base(2)
  set ::GID2NET::LIST_INFO(POINT_PLANE_X,$indice) $point_plane(0)
  set ::GID2NET::LIST_INFO(POINT_PLANE_Y,$indice) $point_plane(1)
  set ::GID2NET::LIST_INFO(POINT_PLANE_Z,$indice) $point_plane(2)
  set ::GID2NET::LIST_INFO(POINT2_X,$indice) $point2(0)
  set ::GID2NET::LIST_INFO(POINT2_Y,$indice) $point2(1)
  set ::GID2NET::LIST_INFO(POINT2_Z,$indice) $point2(2)
  set ::GID2NET::LIST_INFO(RADIO1,$indice) $radio_base
  set ::GID2NET::LIST_INFO(RADIO2,$indice) 0
  set ::GID2NET::LIST_INFO(NORMAL_X,$indice) $normal_base(0)
  set ::GID2NET::LIST_INFO(NORMAL_Y,$indice) $normal_base(1)
  set ::GID2NET::LIST_INFO(NORMAL_Z,$indice) $normal_base(2)
  #almacenar las condiciones de contorno
  set ::GID2NET::LIST_INFO(BC_S2,$indice) $bc_s2
  set ::GID2NET::LIST_INFO(BC_LATERAL,$indice) $bc_lateral
  #almacenar los planos
  if {$bc_s2 == 0} {
    set ::GID2NET::LIST_INFO(PLANE1,$indice) "plane($center_base(0),\
$center_base(1),$center_base(2);$normal_base(0),$normal_base(1),$normal_base(2))"
  } else {
    set ::GID2NET::LIST_INFO(PLANE1,$indice) "plane($center_base(0),\
```

```
$center_base(1),$center_base(2);$normal_base(0),$normal_base(1),$normal_base(2))\
 -bc = $bc_s2"
  }
  set ::GID2NET::LIST_INFO(PLANE2,$indice) "plane ($point_plane(0),$point_plane(1),\
$point_plane(2);[expr (-1)*$normal_base(0)],[expr (-1)*$normal_base(1)],\
[expr (-1)*$normal_base(2)])"
  if {$bc_lateral == 0} {
    set ::GID2NET::LIST_FILE($indice) "cone($center_base(0),$center_base(1),\
$center_base(2);$radio_base;$point2(0),$point2(1),$point2(2);0)"
  } else {
    set ::GID2NET::LIST_FILE($indice) "cone($center_base(0),$center_base(1),\
$center_base(2);$radio_base;$point2(0),$point2(1),$point2(2);0) -bc = $bc_lateral"
  }
}

#Metodo para generar las operaciones booleanas
proc Operation {op} {
  #almacenar la info antes de comenzar
  Button_scape
  #obtener los id de los dos volumenes que se van a operar
  set id_vol_1 [GidUtils::PickEntities volumes single]
  set id_vol_2 [GidUtils::PickEntities volumes single]
  #los volumenes tienen que ser diferentes
  if {$id_vol_1 == $id_vol_2} {
    WarnWin "Seleccione dos volumenes diferentes"
    return
  }
  set id_1 [expr $id_vol_1 - 1]
  set id_2 [expr $id_vol_2 - 1]

  #almacenar el tipo de operacion que se ha realizado
  set ::GID2NET::LIST_OPERATIONS(TYPE,$::GID2NET::NUM_OP) $op

  #Extraemos la info de los volumenes que componen la operacion
  for {set i 0} {$i < 2} {incr i} {
    if {$i==0} {
      set id $id_1
    }
    if {$i==1} {
      set id $id_2
    }
    set type $::GID2NET::LIST_INFO(TYPE,$id)
    #construir las primitivas de las operaciones
    if {$type == 0} {
      set nombre_vol "cube_$id"
      set planes "solid p1_$id = plane($::GID2NET::LIST_INFO(CENTER_S1_X,$id),\
$::GID2NET::LIST_INFO(CENTER_S1_Y,$id),$::GID2NET::LIST_INFO(CENTER_S1_Z,$id);\
$::GID2NET::LIST_INFO(NORMAL_S1_X,$id),$::GID2NET::LIST_INFO(NORMAL_S1_Y,$id),\
$::GID2NET::LIST_INFO(NORMAL_S1_Z,$id)) -bc = $::GID2NET::LIST_INFO(BC_S1,$id);
solid p2_$id = plane($::GID2NET::LIST_INFO(CENTER_S2_X,$id),\
$::GID2NET::LIST_INFO(CENTER_S2_Y,$id),$::GID2NET::LIST_INFO(CENTER_S2_Z,$id);\
$::GID2NET::LIST_INFO(NORMAL_S2_X,$id),$::GID2NET::LIST_INFO(NORMAL_S2_Y,$id),\
$::GID2NET::LIST_INFO(NORMAL_S2_Z,$id)) -bc = $::GID2NET::LIST_INFO(BC_S2,$id);
solid p3_$id = plane($::GID2NET::LIST_INFO(CENTER_S3_X,$id),\
$::GID2NET::LIST_INFO(CENTER_S3_Y,$id),$::GID2NET::LIST_INFO(CENTER_S3_Z,$id);\
$::GID2NET::LIST_INFO(NORMAL_S3_X,$id),$::GID2NET::LIST_INFO(NORMAL_S3_Y,$id),\
$::GID2NET::LIST_INFO(NORMAL_S3_Z,$id)) -bc = $::GID2NET::LIST_INFO(BC_S3,$id);
solid p4_$id = plane($::GID2NET::LIST_INFO(CENTER_S4_X,$id),\
$::GID2NET::LIST_INFO(CENTER_S4_Y,$id),$::GID2NET::LIST_INFO(CENTER_S4_Z,$id);\
$::GID2NET::LIST_INFO(NORMAL_S4_X,$id),$::GID2NET::LIST_INFO(NORMAL_S4_Y,$id),\
$::GID2NET::LIST_INFO(NORMAL_S4_Z,$id)) -bc = $::GID2NET::LIST_INFO(BC_S4,$id);
solid p5_$id =  plane($::GID2NET::LIST_INFO(CENTER_S5_X,$id),\
$::GID2NET::LIST_INFO(CENTER_S5_Y,$id),$::GID2NET::LIST_INFO(CENTER_S5_Z,$id);\
$::GID2NET::LIST_INFO(NORMAL_S5_X,$id),$::GID2NET::LIST_INFO(NORMAL_S5_Y,$id),\
$::GID2NET::LIST_INFO(NORMAL_S5_Z,$id)) -bc = $::GID2NET::LIST_INFO(BC_S5,$id);
solid p6_$id =  plane($::GID2NET::LIST_INFO(CENTER_S6_X,$id),\
$::GID2NET::LIST_INFO(CENTER_S6_Y,$id),$::GID2NET::LIST_INFO(CENTER_S6_Z,$id);\
```

```
$::GID2NET::LIST_INFO(NORMAL_S6_X,$id),$::GID2NET::LIST_INFO(NORMAL_S6_Y,$id),\
$::GID2NET::LIST_INFO(NORMAL_S6_Z,$id)) -bc = $::GID2NET::LIST_INFO(BC_S6,$id);"

      set vol "solid cube_$id = p1_$id and p2_$id and p3_$id and p4_$id and p5_$id and p6_$id;"

    }
    if {$type == 1} {
      set nombre_vol "hemisphere_$id"
      set planes ""
      set vol "solid hemisphere_$id = sphere($::GID2NET::LIST_INFO(CENTER_X,$id),\
$::GID2NET::LIST_INFO(CENTER_Y,$id),$::GID2NET::LIST_INFO(CENTER_Z,$id);\
$::GID2NET::LIST_INFO(RADIO,$id)) -bc = $::GID2NET::LIST_INFO(BC,$id);"
    }
    if {$type == 2} {
      set nombre_vol "cylinder_$id"
      set planes "solid p1_$id = plane($::GID2NET::LIST_INFO(C1_X,$id),\
$::GID2NET::LIST_INFO(C1_Y,$id),$::GID2NET::LIST_INFO(C1_Z,$id);\
$::GID2NET::LIST_INFO(N_S1_X,$id),$::GID2NET::LIST_INFO(N_S1_Y,$id),\
$::GID2NET::LIST_INFO(N_S1_Z,$id)) -bc = $::GID2NET::LIST_INFO(BC_S1,$id);
solid p2_$id = plane($::GID2NET::LIST_INFO(C2_X,$id),$::GID2NET::LIST_INFO(C2_Y,$id),\
$::GID2NET::LIST_INFO(C2_Z,$id);$::GID2NET::LIST_INFO(N_S2_X,$id),\
$::GID2NET::LIST_INFO(N_S2_Y,$id),$::GID2NET::LIST_INFO(N_S2_Z,$id)) \
-bc = $::GID2NET::LIST_INFO(BC_S2,$id);
solid cyl_$id = cylinder($::GID2NET::LIST_INFO(C1_X,$id),$::GID2NET::LIST_INFO(C1_Y,$id),\
$::GID2NET::LIST_INFO(C1_Z,$id);$::GID2NET::LIST_INFO(C2_X,$id),\
$::GID2NET::LIST_INFO(C2_Y,$id),$::GID2NET::LIST_INFO(C2_Z,$id);\
$::GID2NET::LIST_INFO(RADIO,$id)) -bc = $::GID2NET::LIST_INFO(BC_LATERAL,$id);"

      set vol "solid cylinder_$id = cyl_$id and p1_$id and p2_$id;"
    }
    if {$type == 3} {
      set nombre_vol "cone_$id"
      set planes "solid p1_$id = plane($::GID2NET::LIST_INFO(CENTER1_X,$id),\
$::GID2NET::LIST_INFO(CENTER1_Y,$id),$::GID2NET::LIST_INFO(CENTER1_Z,$id);\
$::GID2NET::LIST_INFO(NORMAL_X,$id),$::GID2NET::LIST_INFO(NORMAL_Y,$id),\
$::GID2NET::LIST_INFO(NORMAL_Z,$id)) -bc = $::GID2NET::LIST_INFO(BC_S2,$id);
solid p2_$id = plane($::GID2NET::LIST_INFO(POINT_PLANE_X,$id),\
$::GID2NET::LIST_INFO(POINT_PLANE_Y,$id),$::GID2NET::LIST_INFO(POINT_PLANE_Z,$id);\
[expr (-1)*$::GID2NET::LIST_INFO(NORMAL_X,$id)],[expr (-1)*$::GID2NET::LIST_INFO(NORMAL_Y,$id)],\
[expr (-1)*$::GID2NET::LIST_INFO(NORMAL_Z,$id)]) -bc = 0;
solid c_$id = cone($::GID2NET::LIST_INFO(CENTER1_X,$id),$::GID2NET::LIST_INFO(CENTER1_Y,$id),\
$::GID2NET::LIST_INFO(CENTER1_Z,$id);$::GID2NET::LIST_INFO(RADIO1,$id);\
$::GID2NET::LIST_INFO(POINT2_X,$id),$::GID2NET::LIST_INFO(POINT2_Y,$id),\
$::GID2NET::LIST_INFO(POINT2_Z,$id);0) -bc = $::GID2NET::LIST_INFO(BC_LATERAL,$id);"

      set vol "solid cone_$id = c_$id and p1_$id and p2_$id;
      tlo cone_$id;"
    }
    if {$i==0} {
      set nombre_v1 $nombre_vol
      set ::GID2NET::LIST_OPERATIONS(PLANES1,$::GID2NET::NUM_OP) $planes
      set ::GID2NET::LIST_OPERATIONS(VOL1,$::GID2NET::NUM_OP) $vol
    }
    if {$i==1} {
      set nombre_v2 $nombre_vol
      set ::GID2NET::LIST_OPERATIONS(PLANES2,$::GID2NET::NUM_OP) $planes
      set ::GID2NET::LIST_OPERATIONS(VOL2,$::GID2NET::NUM_OP) $vol
    }
  }
  #realizar la operacion en funcion del tipo seleccionado
  if {$op == 1} {
    #almacenamos el string que se va a imprimir
    set ::GID2NET::LIST_OPERATIONS(INFO,$::GID2NET::NUM_OP) "$nombre_v1 and $nombre_v2"
    #Asignamos la condicion GID de op
    set ::GID2NET::LAST_ID [expr $::GID2NET::LAST_ID + 1]
    #almacena el identificador del volumen de la operacion
    set ::GID2NET::LIST_OPERATIONS(NUM_VOL,$::GID2NET::NUM_OP) $::GID2NET::LAST_ID
```

```
    GiD_Process Mescape Geometry Create IntSolid3D Intersect $id_vol_1 $id_vol_2 escape
    #se asigna la condicion GID para operaciones
    GiD_AssignData condition Type Volumes 4 $::GID2NET::LAST_ID
    #se ha creado una nueva operacion
    set ::GID2NET::NUM_OP [expr $::GID2NET::NUM_OP + 1]
    Button_scape
  }
  if {$op == 2} {
    #almacenamos el string que se va a imprimir
    set ::GID2NET::LIST_OPERATIONS(INFO,$::GID2NET::NUM_OP) "$nombre_v1_or_$nombre_v2"
    #Asignamos la condicion GID de op
    set ::GID2NET::LAST_ID [expr $::GID2NET::LAST_ID + 1]
    #almacena el identificador del volumen de la operacion
    set ::GID2NET::LIST_OPERATIONS(NUM_VOL,$::GID2NET::NUM_OP) $::GID2NET::LAST_ID
    GiD_Process Mescape Geometry Create IntSolid3D Union $id_vol_1 $id_vol_2 escape
    #se asigna la condicion GID para operaciones
    GiD_AssignData condition Type Volumes 4 $::GID2NET::LAST_ID
    #se ha creado una nueva operacion
    set ::GID2NET::NUM_OP [expr $::GID2NET::NUM_OP + 1]
    Button_scape
  }
  if {$op == 3} {
    set ::GID2NET::LIST_OPERATIONS(INFO,$::GID2NET::NUM_OP) "$nombre_v1_and_not_$nombre_v2"
    #creamos la operacion
    GiD_Process Mescape Geometry Create IntSolid3D SubstractAndIntersect \
    $id_vol_1 escape $id_vol_2 escape
    #Asignamos la condicion GID de op
    set ::GID2NET::LAST_ID [expr $::GID2NET::LAST_ID + 1]
    #almacenamos el identificador del primer volumen generado
    #para tener la referencia de la operacion
    set ::GID2NET::LIST_OPERATIONS(NUM_VOL,$::GID2NET::NUM_OP) $::GID2NET::LAST_ID
    GiD_AssignData condition Type Volumes 5 $::GID2NET::LAST_ID
    set ::GID2NET::LAST_ID [expr $::GID2NET::LAST_ID + 1]
    GiD_AssignData condition Type Volumes 6 $::GID2NET::LAST_ID
    #se ha creado una nueva operacion
    set ::GID2NET::NUM_OP [expr $::GID2NET::NUM_OP + 1]
    set ::GID2NET::AND_NOT 1
    Button_scape
  }
}

#Metodo que construye el fichero geo
proc Create_file {} {

  #inicializa variables locales
  set actual_volumes [GiD_Info Geometry NumVolumes]
  set op 0
  #condicion para que el proyecto este guardado antes de crear el fichero
  set name [GiD_Info Project ModelName]
  if {$name == "UNNAMED"} {
    WarnWin "Save_the_current_project"
    return
  } else {
    set file_name [file join $name.gid "Gid2Net.geo"]
    set ::GID2NET::RUTA_GEO $file_name
  }
  #abrir el fichero y comenzar a escribir
  set file_id [open $file_name w+ 0775]
  puts $file_id "algebraic3d"
  for {set i 0} {$i < $::GID2NET::LAST_ID} {incr i} {
    set enc 0
    for {set j 0} {$j < $::GID2NET::NUM_OP} {incr j} {
      if {$::GID2NET::LIST_OPERATIONS(NUM_VOL,$j) == [expr $i + 1]} {
        if {$::GID2NET::LIST_OPERATIONS(BORRADO,$j) == 0} {
          set op [expr $op +1]
          puts $file_id "$::GID2NET::LIST_OPERATIONS(PLANES1,$j)"
          puts $file_id "$::GID2NET::LIST_OPERATIONS(VOL1,$j)"
```

```
            puts $file_id "$::GID2NET::LIST_OPERATIONS(PLANES2,$j)"
            puts $file_id "$::GID2NET::LIST_OPERATIONS(VOL2,$j)"
            puts $file_id "solid op_$op = $::GID2NET::LIST_OPERATIONS(INFO,$j);"
            puts $file_id "tlo op_$op;"
            set enc 1;
          }
        }
      }
      if {$enc == 0} {
        if {$::GID2NET::LIST_INFO(TYPE,$i) == 0} {
          puts $file_id "solid p1_$i = $::GID2NET::LIST_INFO(PLANE1,$i);"
          puts $file_id "solid p2_$i = $::GID2NET::LIST_INFO(PLANE2,$i);"
          puts $file_id "solid p3_$i = $::GID2NET::LIST_INFO(PLANE3,$i);"
          puts $file_id "solid p4_$i = $::GID2NET::LIST_INFO(PLANE4,$i);"
          puts $file_id "solid p5_$i = $::GID2NET::LIST_INFO(PLANE5,$i);"
          puts $file_id "solid p6_$i = $::GID2NET::LIST_INFO(PLANE6,$i);"
          puts $file_id "solid cube_$i = $::GID2NET::LIST_FILE($i);"
          puts $file_id "tlo cube_$i;"
        } elseif {$::GID2NET::LIST_INFO(TYPE,$i) == 1} {
          puts $file_id "solid hemisphere_$i = $::GID2NET::LIST_FILE($i);"
          puts $file_id "tlo hemisphere_$i;"
        } elseif {$::GID2NET::LIST_INFO(TYPE,$i) == 2} {
          puts $file_id "solid p1_$i = $::GID2NET::LIST_INFO(PLANE1,$i);"
          puts $file_id "solid p2_$i = $::GID2NET::LIST_INFO(PLANE2,$i);"
          puts $file_id "solid cyl_$i = $::GID2NET::LIST_FILE($i);"
          puts $file_id "solid cylinder_$i = cyl_$i and p1_$i and p2_$i;"
          puts $file_id "tlo cylinder_$i;"
        } elseif {$::GID2NET::LIST_INFO(TYPE,$i) == 3} {
          puts $file_id "solid p1_$i = $::GID2NET::LIST_INFO(PLANE1,$i);"
          puts $file_id "solid p2_$i = $::GID2NET::LIST_INFO(PLANE2,$i);"
          puts $file_id "solid c_$i = $::GID2NET::LIST_FILE($i);"
          puts $file_id "solid cone_$i = c_$i and p1_$i and p2_$i;"
          puts $file_id "tlo cone_$i;"
        }
      }
    }
    #cerrar fichero
    close $file_id
}

#Metodo que lee el fichero con la informacion del proyecto
proc Read_save_file {project_name} {
    set name $project_name
    set save_file [file join $name "Project_info.txt"]
    set save_id [open $save_file r]
    for {set i 0} {$i < 6} {incr i} {
      set line [gets $save_id]
      if {$i == 2} {
        set ::GID2NET::LAST_ID $line
      }
      if {$i == 4} {
        set ::GID2NET::NUM_OP $line
      }
    }
    while {[gets $save_id line] >= 0} {
      if {$line == "ID_OP"} {
        set line [gets $save_id]
        set op $line
        set line [gets $save_id]
        if {$line == "TYPE"} {
          set line [gets $save_id]
          set ::GID2NET::LIST_OPERATIONS(TYPE,$op) $line
          set line [gets $save_id]
          if {$line == "NUM_VOL"} {
            set line [gets $save_id]
            set ::GID2NET::LIST_OPERATIONS(NUM_VOL,$op) $line
            set line [gets $save_id]
```

```
            if {$line == "PLANES1"} {
              set line [gets $save_id]
              set planes1 $line
              while {$line != "VOL1"} {
                set line [gets $save_id]
                if {$line == "VOL1"} {
                } else {
                  set planes1 "$planes1 $line"
                }
              }
              set ::GID2NET::LIST_OPERATIONS(PLANES1,$op) "$planes1"
              if {$line == "VOL1"} {
                set line [gets $save_id]
                set ::GID2NET::LIST_OPERATIONS(VOL1,$op) "$line"
                set line [gets $save_id]
                if {$line == "PLANES2"} {
                  set line [gets $save_id]
                  set planes2 $line
                  while {$line != "VOL2"} {
                    set line [gets $save_id]
                    if {$line == "VOL2"} {
                    } else {
                      set planes2 "$planes2 $line"
                    }
                  }
                  set ::GID2NET::LIST_OPERATIONS(PLANES2,$op) "$planes2"
                  if {$line == "VOL2"} {
                    set line [gets $save_id]
                    set ::GID2NET::LIST_OPERATIONS(VOL2,$op) "$line"
                    set line [gets $save_id]
                    if {$line == "INFO"} {
                      set line [gets $save_id]
                      set ::GID2NET::LIST_OPERATIONS(INFO,$op) $line
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
  close $save_id
}

#Metodo que construye el fichero con la informacion del proyecto
proc Write_save_file {} {
  global DIR
  set name [GiD_Info Project ModelName]
  set save_file [file join $name.gid "Project_info.txt"]
  set save_id [open $save_file w+ 0775]
  puts $save_id "INFO_PROJECT"
  puts $save_id "#Numero del ultimo identificador de volumen"
  puts $save_id $::GID2NET::LAST_ID
  puts $save_id "#Numero de operaciones"
  puts $save_id $::GID2NET::NUM_OP
  puts $save_id "#Informacion de las operaciones"
  for {set i 0} {$i < $::GID2NET::NUM_OP} {incr i} {
    puts $save_id "ID_OP"
    puts $save_id "$i"
    puts $save_id "TYPE"
    puts $save_id "$::GID2NET::LIST_OPERATIONS(TYPE,$i)"
    puts $save_id "NUM_VOL"
    puts $save_id "$::GID2NET::LIST_OPERATIONS(NUM_VOL,$i)"
    puts $save_id "PLANES1"
    puts $save_id "$::GID2NET::LIST_OPERATIONS(PLANES1,$i)"
    puts $save_id "VOL1"
```

```
    puts $save_id "$::GID2NET::LIST_OPERATIONS(VOL1,$i)"
    puts $save_id "PLANES2"
    puts $save_id "$::GID2NET::LIST_OPERATIONS(PLANES2,$i)"
    puts $save_id "VOL2"
    puts $save_id "$::GID2NET::LIST_OPERATIONS(VOL2,$i)"
    puts $save_id "INFO"
    puts $save_id "$::GID2NET::LIST_OPERATIONS(INFO,$i)"
  }
  close $save_id
}

#Metodo para la asignacion de las condiciones de contorno
proc Assign_bc {bc} {
  global OTRA_BC
  if {$bc == 4} {
    set bc $OTRA_BC
  }
  #el usuario seleeciona la superficie a la que desea asignarle la BC
  set id_surface [GidUtils::PickEntities surfaces single]
  #asignacion de la bc
  GiD_AssignData condition Boundary Surfaces $bc $id_surface
  set ::GID2NET::new_bc 1
  #como ha habido cambios hay que almacenar la informacion
  Button_scape
}

#metodo que muestra por colores las BC de los volumenes
proc View_colors_bc {} {
   GiD_Process Mescape Data Conditions DrawCond -ByColor- "Boundary" "ID"
}

#metodo que muestra con numeros las BC de los volumenes
proc View_numbers_bc {} {
  GiD_Process Mescape Data Conditions DrawCond "Boundary" "ID"
}

#Metodo que crea y carga la malla generada por NetGen
proc Create_meshFile {} {
  global DIR
  #crear el fichero geo
  Create_file
  #escribe el script para la llamada remota
  Escribir_script
  #llamada remota a NetGen
  exec [file join $DIR script.bat]
  #se construye el fichero GiD2Net.msh
  Create_GiD$^{\begin{small}\textregistered\end{small}}$mesh
  #path del fichero msh
  set path_project [GiD_Info Project ModelName]
  set new_msh [file join ${path_project}.gid "Net2Gid.msh"]
  #se importa en GID el mallado
  GiD_Process Mescape Files MeshRead $new_msh
  GiD_Process Mescape Meshing MeshView
  GiD_Process Mescape Files MeshRead $new_msh
  GiD_Process Mescape Meshing MeshView
}

#Metodo que escribe el script para la llamada a NetGen
proc Escribir_script {} {
  global DIR
  #tamanno de malla
  set mesh_size [GiD_Info Project RecommendedMeshSize]
  #crear el fichero
  set file_name [file join $DIR "script.bat"]
  set script_id [open  $file_name w+ 0775]
  set dir_netgen [file join "C:/Archivos_de_programa" \
"Netgen-4.9.13_Win32" "bin" "netgen.exe"]
```

```
  set path_project [GiD_Info Project ModelName]
  set dir_project [file join ${path_project}.gid "GiDtoNetNet.msh"]
  set options "-geofile=\"$::GID2NET::RUTA_GEO\" -batchmode \
-meshfiletype=\"Neutral Format\" -meshfile=\"$dir_project\" -meshsize=$mesh_size"
  puts $script_id "\"$dir_netgen\" $options"
  #cerrar fichero
  close $script_id
}

#Metodo que crea el fichero GiD2Net.msh con la informacion del
#mallado dado por NetGen
proc Create_GiD$^{\begin{small}\textregistered\end{small}}$mesh {} {
  global COORDS_X
  global COORDS_Y
  global COORDS_Z
  #abrir el fichero GiDtoNet.msh
  set path_project [GiD_Info Project ModelName]
  set dir_project [file join ${path_project}.gid "GiDtoNetNet.msh"]
  set file_msh_id [open $dir_project r 0775]
  #nuevo fichero GiD2Net.msh
  set new_msh [file join ${path_project}.gid "Net2Gid.msh"]
  set new_msh_id [open $new_msh  w+ 0775]
  puts $new_msh_id "MESH    dimension 3 ElemType Tetrahedra  Nnode 4"
  puts $new_msh_id "Coordinates"
  set linea [GetLineValues $file_msh_id]
  set num_nodes [lindex $linea 0]
  for {set i 0} {$i < $num_nodes} {incr i} {
    set linea [GetLineValues $file_msh_id]
    set COORDS_X($i) [lindex $linea 0]
    set COORDS_Y($i) [lindex $linea 1]
    set COORDS_Z($i) [lindex $linea 2]
    puts $new_msh_id "  [expr $i + 1]\t[lindex $linea 0]\t[lindex $linea 1]\t[lindex $linea 2]"
  }
  puts $new_msh_id "end coordinates"
  puts $new_msh_id ""
  puts $new_msh_id "Elements"
  set linea [GetLineValues $file_msh_id]
  set num_elem [lindex $linea 0]
  for {set i 0} {$i < $num_elem} {incr i} {
    set linea [GetLineValues $file_msh_id]
    #se realizan las operaciones necesarias para
    #construir el fichero de la manera correcta
    set n1 [lindex $linea 1]
    set n2 [lindex $linea 2]
    set n3 [lindex $linea 3]
    set n4 [lindex $linea 4]
    #se comprueba si hay que reordenar los nodos
    set vect_21 [Calc_vect $n1 $n2]
    set vect_31 [Calc_vect $n1 $n3]
    set vect_41 [Calc_vect $n1 $n4]
    set prod_vect [Cross_product $vect_21 $vect_31]
    set op1 [lindex $prod_vect 0]
    set op2 [lindex $vect_41 0]
    set op3 [lindex $prod_vect 0]
    set op4 [lindex $vect_41 0]
    set op5 [lindex $prod_vect 1]
    set op6 [lindex $vect_41 1]
    set op7 [lindex $prod_vect 1]
    set op8 [lindex $vect_41 1]
    set op9 [lindex $prod_vect 2]
    set op10 [lindex $vect_41 2]
    set op11 [lindex $prod_vect 2]
    set op12 [lindex $vect_41 2]
    set and1 [expr $op1>=0 && $op2>=0]
    set and2 [expr $op3<0 && $op4<0]
    set and3 [expr $op5>=0 && $op6>=0]
    set and4 [expr $op7<0 && $op8<0]
```

```
    set and5 [expr $op9>=0 && $op10>=0]
    set and6 [expr $op11<0 && $op12<0]
    set or1 [expr $and1 || $and2]
    set or2 [expr $and3 || $and4]
    set or3 [expr $and5 || $and6]
    #se escriben las conectividades
    if {$or1 == 1 && $or2 == 1 && $or3 == 1} {
      puts $new_msh_id "\t[expr $i+1]\t\t[lindex $linea 1]\
\t\t[lindex $linea 2]\t\t[lindex $linea 3]\t\t[lindex $linea 4]"
    } else {
      puts $new_msh_id "\t[expr $i+1]\t\t[lindex $linea 2]\
\t\t[lindex $linea 4]\t\t[lindex $linea 1]\t\t[lindex $linea 3]"
    }
  }
  puts $new_msh_id "end_elements"
  #cerrar ambos ficheros
  close $file_msh_id
  close $new_msh_id
}

##############################
# Procedimientos auxliares #
##############################

#Metodo que devuelve el contenido de una linea
#de un fichero en una lista
proc GetLineValues { file } {

   #Inicialize the values list
   set values [list]
   #Read a line of the file
   set line [join [gets $file]]
   #Get the length of the line
   set lngth [string length $line]
   #Loop that it lets get character by character
   for {set pos 0; set val ""} {$pos < $lngth} {incr pos} {
       #Get one character
       set char [string index $line $pos]
       #If char is a space
       if {$char eq " " || $char eq "\n"} {
           #Insert the value in the list
           lappend values $val
           #Inicialize variable val to empty
           set val ""; continue
       }
       #If the char is not a space
       set val ${val}${char}
   }
   #Insert the last value in the list
   lappend values $val
   #Return the list with the values
   return $values
}

#Metodo para realizar el producto vectorial
proc Cross_product  {vector1 vector2} {

  #obtencion de coordenadas de nodos
  set coord1_x [lindex $vector1 0]
  set coord1_y [lindex $vector1 1]
  set coord1_z [lindex $vector1 2]
  set coord2_x [lindex $vector2 0]
  set coord2_y [lindex $vector2 1]
  set coord2_z [lindex $vector2 2]
  set p1 [expr $coord1_y*$coord2_z]
  set p2 [expr $coord1_z*$coord2_y]
  set p3 [expr $coord1_z*$coord2_x]
```

```
  set p4 [expr $coord1_x*$coord2_z]
  set p5 [expr $coord1_x*$coord2_y]
  set p6 [expr $coord1_y*$coord2_x]
  #producto vectorial
  set prod_x  [expr $p1 - $p2]
  set prod_y  [expr $p3 - $p4]
  set prod_z  [expr $p5 - $p6]
  #modulo de la operacion
  set modulo [expr sqrt( $prod_x*$prod_x + $prod_y*$prod_y + $prod_z*$prod_z)]
  set prod [list]
  #almacenamiento de los resultados
  lappend prod [expr $prod_x/$modulo]
  lappend prod [expr $prod_y/$modulo]
  lappend prod [expr $prod_z/$modulo]
  return $prod
}

#Metodo que calcula el vector entre dos nodos
proc Calc_vect {nodo1 nodo2} {
  global COORDS_X
  global COORDS_Y
  global COORDS_Z
  set pos1 [expr $nodo1 - 1]
  set pos2 [expr $nodo2 - 1]
  set vect [list]
  lappend vect [expr $COORDS_X($pos2) - $COORDS_X($pos1)]
  lappend vect [expr $COORDS_Y($pos2) - $COORDS_Y($pos1)]
  lappend vect [expr $COORDS_Z($pos2) - $COORDS_Z($pos1)]
  return $vect
}
```

Bibliografía

[1] *Cubit. Mesh and Generation Toolkit.* User Manual.
URL http://cubit.sandia.gov/cubitprogram.html.

[2] *GiD®* User Manual. URL http://www.gidhome.com.

[3] *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities.* URL http://www.geuz.org/gmsh.

[4] *MesGens.* URL http://meshgen.sourceforge.net.

[5] *NetGen 4.3.* Joachim Schöberl. URL http://www.hpfem.jku.at/netgen.

[6] *TrueGrid.* URL http://www.truegrid.com.

[7] *CIMNE - International Center for Numerical Methods in Engineering.* Barcelona 1987. URL www.cimne.upc.es

[8] Proyecto Fin de Carrera. *GiDtohp: Interfaz Basada en Preprocesador GiD® para Modelado Geométrico con Adaptatividad Automática hp.* Daniel García Doñoro. Universidad Carlos III de Madrid 2008.

[9] *Practical Programming in TCL and TK.* Brent B. Welch and Ken Jones. Prentice Hall PTR. 4st ed., June 2003.

[10] GiD®*: The personal pre and postprocessor.* International Center for Numerical Methods in Engineering (CIMNE). URL http://gid.cimne.upc.es/.